

Appendix A. Assumptions

The following is a list of assumptions made in developing the specification of the routines in the SDP Toolkit described in section 6.

6.2 SDP Toolkit Tools—Mandatory

6.2.1 File I/O Tools

6.2.1.1 Level 0 Science Data Access Tools

PGS_IO_L0_Open()

PGS_IO_L0_GetHeader()

PGS_IO_L0_GetPacket()

- a. Level 0 raw data will be in the form of CCSDS-formatted packets.
- b. Level 0 packets will be time-ordered and duplicate packets will have been removed by EDOS or Pacor/DDF.
- c. Level 0 access routines are designed to operate on physical files, which may not be identical to data granules.
- d. Level 0 data files, with associated file attribute metadata, will come through the Science Data Processing Segment (SDPS) ingest data server and will be pre-staged to a given PGE.
- e. ECS Data Ingest will stage and make available file attribute metadata for each physical Level 0 data file staged to a PGE.
- f. Without changing any physical file data, ECS Data Ingest will perform any granularization of Level 0 data to a form other than as is received from SDPF or EDOS (if this does not correspond to the form required by EOS investigators) prior to the staging of Level 0 data to the PGE.
- g. ECS Data Ingest will perform any EOS investigator required subsetting or combination of Level 0 header and quality information that is necessary as a result of granularizing Level 0 data files prior to the staging of the data to the PGE.
- h. ECS Data Ingest will make information on the orbit number corresponding to each physical Level 0 data file available to the SDP Toolkit through associated metadata.
- i. For each SDPF-generated Data Set File staged to a PGE by ECS Data Ingest, the corresponding SFDU header file will also be staged.

- j. Level 0 data files will be staged to a PGE in the machine native format.
- k. For each staged Level 0 data file, the following file attribute metadata parameters, at a minimum, will be staged and available to a PGE for use in science processing:
 - 1. time tag of 1st packet of staged Level 0
 - 2. time tag of last packet of staged Level 0
 - 3. number of physical Level 0 data files staged
 - 4. start time of Level 0 data as requested by investigators through the planner/scheduler system
 - 5. end time of Level 0 data as requested by investigators through the planner/scheduler system
 - 6. APID of each Level 0 data file, if the Level 0 data files are APID-unique
 - 7. orbit number(s) of the staged Level 0 data
- l. Exact format of Level 0 file structures must be fixed by December '94 in order for the Level 0 access tools to be delivered on schedule.

6.2.1.2 HDF File Access Tools

- a. It is assumed that users will obtain and compile the HDF NCSA libraries on their own and link with the PGS. (HDF distribution is available via anonymous ftp from ftp.ncsa.uiuc.edu, 141.142.20.50.)

6.2.1.4 Metadata

PGS_MET_Init()

- a. A Metadata Configuration File (MCF) will be built around the 'parameter = value' form to provide maximum flexibility. Each metadata element will be fully described in the MCF. This information will be held in memory in a set of linked structures or similar constructs.
- b. The core metadata descriptions will be supplied by ECS.
- c. It is assumed that only one header will be initiated at any one time during processing.

PGS_MET_Write()

- a. It is assumed that the output of the metadata tools will be to an HDF formatted product. In each case the product/file may be existing or new. It is assumed that these products/files will be opened and closed using the appropriate tools (e.g., open/close generic file); i.e., the _MET_ tools do not perform these functions.
- b. It is assumed that further interaction with the inventory is done using other software that interacts with the metadata file produced by this tool.

PGS_MET_GetPCAttr()

- a. It is assumed that input products are accessed through the PCF and associated tools
- b. It is assumed that the metadata in input files is available either 1. in the same form as that written by PGS_MET_Write or 2. in a simple separate ASCII text file. In both cases, the metadata file is referenced in the field prescribed by the PCF rules.

PGS_MET_GetConfig ()

- a. It is assumed that configuration data is held as prescribed by the PCF rules.
- b. It is assumed that configuration data will be accessed using the label field.

6.2.2 Error/Status Reporting Tools

- a. It is assumed that only three log files will need to be created by the Toolkit: Status Message Log, User Status Log and Status Report Log.
- b. Every call to a PGS_SMF_Set* routine results in a status message being appended to the Status Log file.
- c. Status Report entries are directed to the Status Report Log file.
- d. User Status entries are directed to the User Status Log file.

PGS_SMF_SetHDFMsg()

- a. It is assumed that calls to HDF-EOS library routines will set or return an error code and message that can be retrieved by this function for later recall by other error reporting tools, or that the HDF-EOS library will incorporate the existing SMF library calls thereby circumventing the need for this tool.

PGS_SMF_GetActionByCode()

- a. It is assumed that the user only requires the specification and retrieval of an action string, for use in reporting, and not the specification and execution of action methods.

PGS_SMF_CreateMsgTag()

- a. Assumption is that this tool will have access to production run id and science software program id during runtime; thus enabling this routine to generate a unique string based on product id.

PGS_SMF_GenerateStatusReport()

- a. It is assumed that the Toolkit development team has the license to determine the format of the individual status report entries. The format that we have adopted calls for a system-defined message tag to precede a user-provided message string; separators will be inserted between individual report entries for the sake of clarity.

- b. It is assumed that the generation of a status report results in the report being entered into a Status Report Log file created by the Toolkit.

PGS_SMF_SendRuntimeData()

- a. It is assumed that this toolkit will interface to some other toolkit, or Communications and Systems Management Segment (CSMS) functionality, to effect the transfer of the selected Runtime files to an intermediate holding location. The same mechanism will perform the transmission of one or more e-mail notices to alert the interested parties as to the disposition of the Runtime files.
- b. It is also assumed that there will be a defined intermediate holding location for this toolkit to send the Runtime files at the DAAC site and that there will be an interface to alert the monitoring authority that these Runtime files have arrived.

6.2.3 Process Control Tools

- a. a PGE process control database record will exist as a UNIX file or Database Management System (DBMS) record for each PGE within the DAAC.
- b. A template PGE process control database record will be "seeded" with user-defined information during the integration and testing process.
- c. An instance of the PGE process control database record will be populated with the appropriate runtime data and if necessary, staged prior to PGE execution.
- d. Runtime parameter values may be modified prior to runtime through some as yet unidentified interface/mechanism.
- e. A one-to-many logical-to-physical file relationship may exist for input products.
- f. The Planning & Data Production System (PDPS) will provide for Toolkit initialization allowing internal Toolkit structures to become populated.
- g. The PDPS will provide for Toolkit termination, allowing the Toolkit to perform necessary housekeeping and ensuring that important intermediate data gets saved for future runs of the same PGE.

PGS_PC_GenUniqueID()

- a. It is assumed that the Science Software Program ID and the Production Run ID are system defined values that will be available from the execution environment, or from the PGE process control database during Toolkit Initialization.
- b. The logical Product ID value passed in by the user will be defined by the user, but will have been mapped to a DAAC-based intermediate identifier during the Integration & Test phase

PGS_PC_GetConfigData()

PGS_PC_GetConfigDataCom

- a. Each user-defined logical Runtime Parameter ID passed into this function will be mapped to an actual runtime parameter during I&T. This will allow the Parameter ID to be resolved into a default value, or an overriding value at runtime.

PGS_PC_GetReference()

- a. It is assumed that users of HDF will utilize this tool to obtain a reference to pass to the HDF open library call.

PGS_PC_GetNumberOfFiles()

PGS_PC_GetNumberOfFilesCom

- a. To satisfy the one-to-many logical-to-physical file relationship, the user, upon retrieving the number of files per given identifier with this tool, will be able to index to the desired instance of a file by providing the version number to the appropriate file I/O toolkit function.

PGS_PC_GetFileAttr()

PGS_PC_GetFileByAttr()

PGS_PC_GetFileAttrCom

- a. It is assumed that input product metadata and file attributes will be made directly available to the Toolkit through the PGE Process Control Database.
- b. If available, it is assumed that input support file metadata and file attributes will be made directly available to the Toolkit through the PGE Process Control Database.

6.2.4 Memory Management Tools

Dynamic Memory Tools

- a. It is assumed that all dynamic memory allocated within the user's program is obtained through the use of these tools.

Shared Memory Tools

- a. One basic assumption is that all the executables will be invoked within a shell script (i.e., PGE).
- b. Additionally, that there will be a shell script that wraps around the main PGE shell script, allowing an initialization program to create a shared memory segment for the Toolkit; this will enable the Toolkit to facilitate tracking of all the necessary resources needed to support shared memory capabilities for the user. That same shell script will allow a termination program to release all the shared-memory resources used by both the Toolkit and the user.

- c. Modification to the existing shared memory API will be minimal if and when the POSIX implementation is adopted.
- d. Shared memory segments will be large enough to support the needs of both the user and the Toolkit.
- e. Two segments, one for the user and one for the Toolkit, can be attached concurrently within the same process.

6.2.5 Bit Manipulation Tools

- a. It is assumed that bit-manipulation functionality will be provided inherently by the language for 'C' and Fortran90, and that users of Fortran77 will use compilers that conform to MIL STD 1753 in order to obtain these capabilities.

6.2.6 Spacecraft Ephemeris and Attitude Data Access Tools

PGS_EPH_EphemAttit()

- a. The specification for reliability of orbit and attitude data is assumed to be provided by Goddard Space Flight Center (GSFC)/Flight dynamics Facility (FDF).
- b. This tool does not compute instrument attitude.
- c. Time is assumed to be input in ASCII time code A or B format.

6.2.7 Time and Date Conversion Tools

PGS_TD_UTCtoTAI()

- a. The current leapseconds file must be available.

PGS_TD_TAItoUTC()

- a. The current leapseconds file must be available.

PGS_TD_UTCtoGPS()

- a. The current leapseconds file must be available.

PGS_TD_GPSstoUTC()

- a. The current leapseconds file must be available.

PGS_TD_Sctime_to_UTC()

- a. The Spacecraft time difference file or coefficients for interpolation must be available. The current leapseconds file must be available.

PGS_TD_UTC_to_SCtime()

- a. The Spacecraft time difference file or coefficients for interpolation must be available. The current leapseconds file must be available. User responsibility to work with difference from nearest tick (interpolate between ticks if desired). It is assumed that this requirement is intended for cross checking of data and that the usual transformation is from Spacecraft Clock time to other standards, such as UTC. If user wants to interpolate, she/he will have to take answer back to UTC and find the difference from the original UTC; then go to next tick on that side and interpolate between the two. It would be possible to rework this tool to provide the two nearest ticks on either side of the UTC time and interpolation weights.

PGS_TD_TimeInterval()

- a. It is user responsibility to supply TAI times, although GPS times can be used instead. The two must not be mixed. All the function does is to subtract double precision numbers.

6.3 SDP Toolkit Tools - Optional

6.3.1 Ancillary Data Access and Manipulation Tools

PGS_AA_dcw()

- a. It is assumed that for access to areas or multiple points, that the user will provide the lat/long coordinates to this tool; i.e., the tool does not include the functionality to calculate other coordinates than those supplied by the user.

PGS_AA_dem()

- a. It is assumed that DEMs will be in raster format.
- b. All assumptions under **PGS_AA_2DRead()** and **PGS_AA_2Dgeo()** apply.

PGS_AA_PeVA()

- a. It is assumed that a large number of static files holding data associated with various algorithms will be in ASCII format. It is further assumed that some of these files will be in the parameter = value format.

PGS_AA_2DRead() and PGS_AA_2Dgeo()

- a. It is assumed that the ancillary data have been prepared into formats suitable for use with this tool; i.e., they are in 2D grids containing data values organized in a raster format and describable using a standard set of metadata.
- b. It is assumed that the ancillary data files will exist as a series of time specific physical files with a clear time-tag (e.g., in the file name); i.e., each physical file contains a full set of the data in spatial terms (e.g., sea ice for one week for the region north of 60 degrees).

- c. It is assumed that for most purposes, a 2 dimensional array of sufficient size can be created to service user requirements.

PGS_AA_3DRead() and PGS_AA_3Dgeo()

- a. It is assumed that the ancillary data have been prepared into formats suitable for use with this tool; i.e., they are in 3D grids containing data values organized in a raster format and describable using a standard set of metadata.
- b. It is assumed that the ancillary data files will exist as a series of time specific physical files with a clear time-tag (e.g., in the file name); i.e., each physical file contains a full set of the data in spatial terms.
- c. It is assumed that for most purposes, a 3 dimensional array of sufficient size can be created to service user requirements.

PGS_AA_INTERP()

This functionality is now part of PGS_AA_2Dgeo. See section D.3.2.3

6.3.2 Celestial Body Position

6.3.2.1 Celestial Body Access Tools

PGS_CBP_Earth_CB_Vector()

- a. Sun, moon, and planetary ephemerides are assumed to exist in an external file.
- b. Time is assumed to be input in CCSDS ASCII time code A or B format.

PGS_CBP_Sat_CB_Vector()

- a. Sun, moon, and planetary ephemerides are assumed to exist in an external file.
- b. Time is assumed to be input in CCSDS ASCII time code A or B format.
- c. Spacecraft ephemeris is assumed to be available in an external file.
- d. Earth to Celestial Body ECI vector is assumed to be computed using the tool of that name.

PGS_CBP_SolarTimeCoords()

- a. Time is assumed to be input in ASCII time code A or B format.

PGS_CBP_body_inFOV()

- a. Sun, moon, and planetary ephemerides are assumed to exist in an external file.
- b. Star locations are assumed to be read from the mission star catalog file received from FDF.

- c. A set of vectors defining the FOV in spacecraft coordinates is assumed to be provided by the user. The vectors must be in sequential order around the FOV periphery.
- d. Time is assumed to be input in ASCII time code A or B format.
- e. Spacecraft ephemeris is assumed to be available in an external file.

PGS_CBP_BrightStar_positions()

- a. Star locations are assumed to be read from the mission star catalog file.
- b. The star catalog is assumed to be created based on a minimum star magnitude TBD by the project.
- c. Time is assumed to be input in ASCII time code A or B format.

6.3.3 Coordinate System Conversion

6.3.3.1 Coordinate System Conversion - Transformation Tools

6.3.3.2 Coordinate System Conversion - Other Tools

PGS_CSC_DayNight()

- a. The position of the sun is assumed to be obtained from the sun, moon, and planetary ephemerides external file.
- b. Time is assumed to be input in CCSDS ASCII time code A or B format.

PGS_CSC_GreenwichHour()

- a. A file of UT1–UTC times is assumed to be present.
- b. Time is assumed to be input in CCSDS ASCII time code A or B format.

PGS_CSC_SubSatPoint()

- a. Time is assumed to be input in CCSDS ASCII time code A or B format.
- b. Spacecraft ephemeris is assumed to be available in an external file.
- c. Earth oblateness model is assumed to be the same as that used to compute the spacecraft ephemeris originally.
- d. A file of UT1–UTC times and Earth polar motion is assumed to be present.

PGS_CSC_Earthpt_FOV()

- a. A set of vectors defining the FOV in spacecraft coordinates is assumed to be provided by the user. The vectors must be in sequential order around the FOV periphery.
- b. Time is assumed to be input in CCSDS ASCII time code A or B format.

- c. Spacecraft ephemeris is assumed to be available in an external file.
- d. Earth oblateness model is assumed to be the same as that used to compute the spacecraft ephemeris originally.
- e. User must supply one vector inside FOV—preferably near center

6.3.4 Geo-Coordinate Transformation Tools

- a. It is assumed that the user has knowledge of the values of the necessary initialization parameters or uses those from the CUC tools (where available).

6.3.6 Constants and Unit Conversions

- a. It is assumed that the constants in this section are supplied by ESDIS.

Appendix B. Status Message File (SMF) Creation and Usage Guidelines

B.1 Note

For a much more simplified explanation about SMF Creation and Usage Guidelines, refer to the SDP Toolkit Primer. The Primer is available on the World Wide Web (WWW). The Universal Research Locator (URL) for the ECS Data Handling System (EDHS) home page is:

<http://edhs1.gsfc.nasa.gov/>

This appendix provides a more detailed description of how Status Message Files (SMFs) are created along with some guidelines on their usage within the science software. Additionally, some examples are provided at the end of this appendix to better illustrate how the software may be used.

B.2 Description

In EOS, messages to the user should be developed using the Status Message File tool set. Together, these tools provide the means to store messages in files that are accessed at runtime to retrieve context-specific message text. Since text messages are stored in runtime files, messages may be modified without recompiling the program that uses the messages. The basic procedure for using these tools follows:

- Create a Status Message File (SMF) that maps status message text to a status label. Additionally, the user may create action message text which maps to the same status label, though this is optional.
- Compile the SMF using the 'smfcompile' program to generate the runtime message file and language-specific "include" file. The runtime message file is used to hold the message/action text. The language-specific "include" file maps the status labels to numeric status numbers via language-specific constructs.
- Use PGS_SMF_Set* tools to preserve a specific status condition.
- Use PGS_SMF_Get* tools to retrieve messages/actions based on the status labels returned by previously called functions.

SMFs require a seed number that is used to generate message/action numbers for message/action labels. This seed number is the key to determining the proper runtime message file and must be unique for each message file. Users cannot simply use any seed number they wish to; they have to be requested and/or assigned by the PGS Toolkit development team. Currently we can support seed numbers up to $(2^{19})-1$ (i.e., 524287). To help identify the proper runtime message file, all message files will be located in a

common message directory, located by the environment variable PGSMSG. This directory will be created by the Toolkit install facility and updated during an smf make procedure.

New updates to this directory may be performed by compiling an SMF text file in the message directory. A more advisable approach would be to maintain each SMF text file in the same directory as the code that relies on the messages contained in the SMF text file. Then compilation of the SMF text file(s) could be setup to precede compilation of the source code (e.g., make smf; make code).

Status Message text file names can be of any valid UNIX filename characters; they must however include a '.t' extension. The generated runtime ASCII message file will be named as PGS_<seed#>, (e.g., PGS_255). The resulting "include" file follows the convention PGS_<tool-group>_<seed#>.[haf] (e.g., PGS_IO_1.h & PGS_IO_1.f). The token <tool-group> is extracted from the 'LABEL' field contained in the SMF text file. For this reason, it would be advisable to name SMF text files with some portion of this field in order to maintain some relationship between the original text file and the smf generated files. To provide a consistent method of status returns, the following procedures should be followed for all software developed for EOS:

- All functions should return one of the following return codes as defined in PGS_SMF.h (FORTRAN users refer to PGS_SMF.f) to indicate the status of the Toolkit operation, unless the function returns a user-defined status as defined in an SMF, or unless a return is unwarranted altogether as in a simple mathematical function (e.g., $y = \text{sine}(x)$):

| | |
|---------------|--|
| PGS_S_SUCCESS | Successful operation |
| PGS_E_ECS | A general ECS error occurred |
| PGS_E_TOOLKIT | A general TOOLKIT error occurred |
| PGS_E_UNIX | A UNIX error occurred |
| PGS_E_HDF | An HDF-EOS error occurred |
| PGS_E_DCE | A DCE error occurred |
| PGS_E_ENV | A Toolkit environment error was detected |

Note that additional defined return codes will be added for various COTS/modules in the future should the need arise.

- Before returning a status code, the unit (i.e., routine, function, procedure, etc.,) should load the specific status information into the static buffer. This is accomplished by calling one of the PGS_SMF_Set* tools.
- The calling function should check the return status of the called unit. If an error condition occurred, the specific error data can be retrieved using the PGS_SMF_Get* tools.

The tools that set or retrieve status data to/from the static buffer area are listed under PGS Error/Status Reporting Tools in the Toolkit User's Guide.

SMF syntax: Syntax for SMF definition is specified in the variant Bachus–Nauer Form (BNF) notation that follows:

BNF notes : [optional item]; { range bounded}; + concatenation [] and space symbols indicate blank or space character

```

allowed_ascii_char ::= {  [! " # & ' ( ) % * + , - . /]
                        [DIGIT]
                        [: ; < = > ? @]
                        [UPPER_CASE_LETTER]
                        [LOWER_CASE_LETTER]
                        [[ \ ] ^ _ ` { | } ~] }

spacing             ::= {[\n] [\t] [ ]}
comment_str        ::= #
instrument          ::= 3{[UPPER_CASE_LETTER]}10
label              ::= 3{[UPPER_CASE_LETTER]}10
level              ::= S | M | U | N | W | E | F
mnemonic           ::= 1{[DIGIT][_][UPPER_CASE_LETTER]}31
mnemonic_label     ::= label + _ + level + _ + mnemonic
action_label       ::= label + _ + A + _ + mnemonic
message_str        ::= 1{[ ] [allowed_ascii_char]}240
action_str         ::= message_str
status_definition  ::= mnemonic_label + spacing +
                        message_str
                        [+ :: + action_label]
action_definition  ::= action_label + spacing + action_str

```

Note on levels:

```

_S_   stands for success
_A_   stands for action (action_label definition only)
_M_   stands for message
_U_   stands for user information
_N_   stands for notice
_W_   stands for warning
_E_   stands for error
_F_   stands for fatal

```

It is up to the user to use the appropriate level in their definition of mnemonics that represent message/action strings. So if an action string is required, use the `_A_` sequence in the `action_label`; if it is an informational–message string use the `_M_` sequence in the `mnemonic_label`; if it is a fatal message string use `_F_` in the `mnemonic_label`. Only `action_labels` use an action level character; the rest of your `mnemonic_label` definitions should use other level characters.

This page intentionally left blank.

Appendix C. Process Control Files

NOTE:

The Master Template PCF as delivered with the Toolkit and described in section C.1.4, MUST be used in its entirety as a template for user PCFs. Please add to it, but do not alter any entries now in it. This file has been populated with dependency information required for proper operation of the Toolkit.

For a much more simplified explanation about Process Control Files and usage, refer to the SDP Toolkit Primer. The Primer is available on the World Wide Web (WWW). The Universal Reference Locator (URL) for the ECS Data Handling System (EDHS) home page is:

<http://edhs1.gsfc.nasa.gov>

This appendix provides a detailed description of how to define and validate Process Control Files.

C.1 Defining Process Control Files

This section of the appendix discusses the various components of a Process Control File (PCF). A sample PCF format is provided as well as an example, which contains the actual entries required to support the Toolkit release 5.1.1.

C.1.1 PCF Components

- **Subject Fields** A process control file MUST contain the following subject fields in the order shown:
 - System Runtime Parameters - unique identifiers used to track instances of a PGE run, versions of science software, etc.
 - Product Input Files - list of ECS standard product data files required as input to the PGE
 - Product Output Files - list of ECS standard product data files generated by the PGE
 - Support Input Files - list of ECS, or Instrument ancillary/support data files required as input to the PGE
 - Support Output Files - list of ECS, or Instrument ancillary/support data files generated by the PGE
 - User Defined Runtime - list of user-defined configuration parameters; Parameters to be accessed by the PGE at runtime

| | | |
|---|---|--|
| Intermediate Input | - | list of non-volatile temporary files required as input to the PGE |
| Intermediate Output | - | list of non-volatile temporary files generated by the PGE |
| Temporary I/O | - | list of volatile temporary files generated and accessed by the PGE at runtime only |
| End | - | PCF terminus |
| <ul style="list-style-type: none"> • Record Fields Each dependency record MUST contain, in the proper order, all of the fields required for the particular type of Subject. | | |
| Identifier | - | Numeric representation of logical identifier (range 10,000–10,999 reserved for Toolkit use only) |
| Reference | - | UNIX file/directory name |
| Path | - | UNIX directory path; start paths with '~' to specify relative paths from \$PGSHOME |
| Reserved | - | Placeholder for future use |
| Universal | - | Universal Reference identifier - may be any string and may contain spaces |
| Attribute | - | Full UNIX path to Product Attribute file |
| Sequence | - | Number of associated Product Input files to follow (inclusive); typically = 1 |
| Description | - | Annotation for parameter; not used in processing |
| Value | - | Assignment to be used during processing; string representation returned by tools |

C.1.2 Format Rules

- All Subject fields are placed in the order shown above
- Each subject field must begin with the question mark token '?'
- The default location entry, for a subject field, must begin with the bang token '!'; there may be only one such entry per subject field and it must immediately follow the subject field declaration.
- All comments must begin with the pound sign token '#'
- Subject and comment tokens must be placed in column one

- There can be no blank lines in the file
- All Record entries must begin in column one
- All Record fields must be delimited with a pipe token '|'
- The last line of the file must begin with a subject field token '?'

C.1.3 Format Example

```
# Process Control Information File
#
#   The Environment variable PGS_PC_INFO_FILE must point to this file.
#   Required inputs appear in bold; all delimiters required.
#   'Path' obtained
#   from the default location entry unless explicitly defined for the
#   individual record.
#
?   SYSTEM RUNTIME PARAMETERS
# -----
# Production Run ID - unique production instance identifier (SCF=1)
# -----
Value
# -----
# Software ID - unique software configuration identifier          (SCF=1)
# -----
Value
#
?   PRODUCT INPUT FILES
!  ~/runtime
#
# -----
# Sequence number must be ordered in a descending fashion
# Ex.
# 100|Instr_Product1A_1.dat|/usr/data||Product1A 1|/usr/data/prod_1A_1.att|3
# 100|Instr_Product1A_2.dat|/usr/data||Product1A 2|/usr/data/prod_1A_2.att|2
# 100|Instr_Product1A_3.dat|/usr/data||Product1A 3|/usr/data/prod_1A_3.att|1
#
# Attribute file MUST reside in same directory as Reference file
# -----
Identifier|Reference|Path|Reserved|Universal|Attribute|Sequence
#
?   PRODUCT OUTPUT FILES
!  ~/runtime
#
```

```

# -----
# Sequence number must be ordered in a descending fashion
# Attribute file MUST reside in same directory as Reference file
# -----
Identifier|Reference|Path|Reserved|Universal|Attribute|Sequence
#
?   SUPPORT INPUT FILES
!  ~/runtime
#
# -----
# Sequence number = 1;
# Attribute file MUST reside in same directory as Reference file
# -----
Identifier|Reference|Path|Reserved|Universal|Attribute|Sequence
#
?   SUPPORT OUTPUT FILES
!  ~/runtime
#
# -----
# Sequence number = 1;
# Attribute file MUST reside in same directory as Reference file
# -----
Identifier|Reference|Path|Reserved|Universal|Attribute|Sequence
#
?   USER DEFINED RUNTIME PARAMETERS
#
# -----
# Value may contain white-space but must be limited to current line;
# Value returned by Toolkit in string representation
# -----
Identifier|Description|Value
#
?   INTERMEDIATE INPUT
!  ~/runtime
#
# -----
# Sequence number = 1;
# Records obtained from INTERMEDIATE OUTPUT field of previous runs
# -----
Identifier|Reference|Path|Reserved|Universal|Reserved|Sequence
#
?   INTERMEDIATE OUTPUT
!  ~/runtime
#

```

```

# -----
# Sequence number = 1;
# Records generated by Toolkit ONLY!
# -----
Identifier | Reference | Path | Reserved | Universal | Reserved | Sequence
#
?   TEMPORARY I/O
!  ~/runtime
#
# -----
# Sequence number = 1;
# Records generated by Toolkit ONLY!
# -----
Identifier | Reference | Path | Reserved | Reserved | Reserved | Sequence
#
?   END

```

C.1.4 Master Template:

The following file was delivered along with the Toolkit Installation. To access this file, set the environment variable PGS_PC_INFO_FILE to '\$PGSHOME/runtime/PCF.relA'.

Initially, this file has been populated with dependency information required for proper operation of the Toolkit. As such, **this file should be considered as a MASTER PCF file from which user PCF files are derived.** To safeguard against the possibility of corrupting essential Toolkit entries, users should use copies of this file as the basis for creating their own. Once a new PCF file has been created, reset the environment variable PGS_PC_INFO_FILE to point to the new file. The new file should now contain all the essential User and Toolkit dependency information. Before using the new PCF, please validate it using the 'pccheck.sh' utility that is located in \$PGSHOME/bin. The effort spent doing so will more than offset the time spent trying to debug the PCF from the errors received while running your program(s). Refer to Part II of this Appendix to see an example on the usage of the 'pccheck.sh' PCF validation tool.

```

#
# filename:
#
#   PCF.relA
#
# description:
#
#   Process Control File (PCF)
#
# notes:

```

```

#
#   This file supports the Rel A version of the toolkit.
#
#   Please treat this file as a master template and make copies of it
#   for your own testing. Note that the Toolkit installation script sets
#   PGS_PC_INFO_FILE to point to this master file by default. Remember
#   to reset the environment variable PGS_PC_INFO_FILE to point to the
#   instance of your PCF.
#
# -----
#   SYSTEM RUNTIME PARAMETERS
# -----
# Production Run ID - unique production instance identifier
# -----
1
# -----
# Software ID - unique software configuration identifier
# -----
1
#
#   PRODUCT INPUT FILES
# Next non-comment line is the default location for PRODUCT INPUT FILES
# WARNING! DO NOT MODIFY THIS LINE unless you have relocated these
# data set files to the location specified by the new setting.
! ~/runtime
#
# -----
# These are actual ancillary data set files - supplied by ECS or the user
# the following are supplied for purposes of tests and as a useful set of

```

ancillary data.

10780|usatile12|||10751|12

10780|usatile11|||10750|11

10780|usatile10|||10749|10

10780|usatile9|||10748|9

10780|usatile8|||10747|8

10780|usatile7|||10746|7

10780|usatile6|||10745|6

10780|usatile5|||10744|5

10780|usatile4|||10743|4

10780|usatile3|||10742|3

10780|usatile2|||10741|2

10780|usatile1|||10740|1

10951|mowel3a.img|||1

10952|owel3a.img|||1

10953|owel14d.img|||1

10954|owel14dr.img|||1

10955|etop05.dat|||1

10956|fnocazm.img|||1

10957|fnococm.img|||1

10958|fnocpt.img|||1

10959|fnocrdg.img|||1

10960|fnocst.img|||1

10961|fnocurb.img|||1

10962|fnocwat.img|||1

10963|fnocmax.imgs|||1

10964|fnocmin.imgs|||1

10965|fnocmod.imgs|||1

```

10966|srzarea.img||||1
10967|srzcode.img||||1
10968|srzphas.img||||1
10969|srzslop.img||||1
10970|srzsoil.img||||1
10971|srztext.img||||1
10972|nmcRucPotPres.datrepack||||1
10973|tbase.bin||||10915|1
10974|tbase.br||||10919|4
10974|tbase.bl||||10918|3
10974|tbase.tr||||10917|2
10974|tbase.tl||||10916|1
10975|geoid.dat||||1
#
# -----
# The following are for the PGS_GCT tool only.
# The IDs are #defined in the PGS_GCT.h file
# -----
10200|nad27sp|~/runtime||||1
10201|nad83sp|~/runtime||||1
# -----
# The following are for the PGS_AA_DCW tool only.
# The IDs are #defined in the PGS_AA_DCW.h file
# -----
10990|eurnesia/||||1
10991|noamer/||||1
10992|soamafr/||||1
10993|sasaus/||||1
#

```

```

# -----
# file for Constant & Unit Conversion (CUC) tools
# IMPORTANT NOTE: THIS FILE WILL BE SUPPLIED AFTER TK4 DELIVERY!
# -----
10999|PGS_CUC_maths_parameters|~/runtime|||1
#
#
#-----
# Metadata Configuration File (MCF) is a template to be filled in by the
Instrument
# teams. The data dictionary is a set of the core metadata attributes and
their
# descriptions. Product specific metadata attribute values need to be added
by the Instrument
# Teams.
#-----
10250|MCF|~/runtime|||1
10251|data_dictionary|~/runtime|||1
#
#
? PRODUCT OUTPUT FILES
# Next line is the default location for PRODUCT OUTPUT FILES
! ~/runtime
#
#
? SUPPORT INPUT FILES
# Next line is the default location for SUPPORT INPUT FILES
! ~/runtime
#
#

```

```

# -----
# This ID is #defined in PGS_AA_Tools.h
# This file contains the IDs for all support and format files shown above
# -----
10900|indexFile|~/runtime|||1
#
# -----
# These are support files for the data set files - to be created by user
# (not necessar
# The IDs must correspond to the logical IDs in the index file
# -----
10901|mowel13aSupport|~/runtime|||1
10902|owel13aSupport|~/runtime|||1
10903|owel14Support|~/runtime|||1
10904|etop05Support|~/runtime|||1
10905|fnoc1Support|~/runtime|||1
10906|fnoc2Support|~/runtime|||1
10907|zobler1Support|~/runtime|||1
10908|zobler2Support|~/runtime|||1
10909|nmcRucSupport|~/runtime|||1
10915|tbaseSupport|~/runtime|||1
10916|tbase1Support|~/runtime|||1
10917|tbase2Support|~/runtime|||1
10918|tbase3Support|~/runtime|||1
10919|tbase4Support|~/runtime|||1
10740|usatile1Support|~/runtime|||1
10741|usatile2Support|~/runtime|||1
10742|usatile3Support|~/runtime|||1
10743|usatile4Support|~/runtime|||1

```

```

10744|usatile5Support|~/runtime|||1
10745|usatile6Support|~/runtime|||1
10746|usatile7Support|~/runtime|||1
10747|usatile8Support|~/runtime|||1
10748|usatile9Support|~/runtime|||1
10749|usatile10Support|~/runtime|||1
10750|usatile11Support|~/runtime|||1
10751|usatile12Support|~/runtime|||1
10948|geoidSupport|~/runtime|||1
#
# -----
# The following are format files for each data set file
# (not necessarily a one-to-one relationship)
# The IDs must correspond to the logical IDs in the index file
# -----
10920|mowe13a.bfm|~/runtime|||1
10921|owe13a.bfm|~/runtime|||1
10922|owe14d.bfm|~/runtime|||1
10923|owe14dr.bfm|~/runtime|||1
10924|etop05.bfm|~/runtime|||1
10925|fnocAzm.bfm|~/runtime|||1
10926|fnocOcm.bfm|~/runtime|||1
10927|fnocPt.bfm|~/runtime|||1
10928|fnocRdg.bfm|~/runtime|||1
10929|fnocSt.bfm|~/runtime|||1
10930|fnocUrb.bfm|~/runtime|||1
10931|fnocWat.bfm|~/runtime|||1
10932|fnocMax.bfm|~/runtime|||1
10933|fnocMin.bfm|~/runtime|||1

```

10934|fnocMod.bfm|~/runtime|||1
10935|srzArea.bfm|~/runtime|||1
10936|srzCode.bfm|~/runtime|||1
10937|srzPhas.bfm|~/runtime|||1
10938|srzSlop.bfm|~/runtime|||1
10939|srzSoil.bfm|~/runtime|||1
10940|srzText.bfm|~/runtime|||1
10941|nmcRucSigPotPres.bfm|~/runtime|||1
10942|tbase.bfm|~/runtime|||1
10943|tbase1.bfm|~/runtime|||1
10944|tbase2.bfm|~/runtime|||1
10945|tbase3.bfm|~/runtime|||1
10946|tbase4.bfm|~/runtime|||1
10700|usatile1.bfm|~/runtime|||1
10701|usatile2.bfm|~/runtime|||1
10702|usatile3.bfm|~/runtime|||1
10703|usatile4.bfm|~/runtime|||1
10704|usatile5.bfm|~/runtime|||1
10705|usatile6.bfm|~/runtime|||1
10706|usatile7.bfm|~/runtime|||1
10707|usatile8.bfm|~/runtime|||1
10708|usatile9.bfm|~/runtime|||1
10709|usatile10.bfm|~/runtime|||1
10710|usatile11.bfm|~/runtime|||1
10711|usatile12.bfm|~/runtime|||1
10947|geoidbfm|~/runtime|||1

```

# leap seconds (TAI-UTC) file
# -----
10301|leapsec.dat|~/database/sun5/TD|||1
#
# -----
# polar motion and UTC-UT1 file
# -----
10401|utcpole.dat|~/database/sun5/CSC|||1
#
# -----
# earth model tags file
# -----
10402|earthfigure.dat|~/database/sun5/CSC|||1
#
# -----
# directory where spacecraft ephemeris files are located
# NOTE: This line is used to specify a directory only!
#       The "file" field should not be altered.
# -----
10501|. |~/database/sun5/EPH|||1
#
# -----
# JPL planetary ephemeris file (binary form)
# -----
10601|de200.eos|~/database/sun5/CBP|||1
#
#
?   SUPPORT OUTPUT FILES
# Next line is default location for SUPPORT OUTPUT FILES

```

```

! ~/runtime
#
#
# -----
# These files support the SMF log functionality. Each run will cause
# status information to be written to 1 or more of the Log files. To
# simulate DAAC operations, remove the 3 Logfiles between test runs.
# Remember: all executables within a PGE will contribute status data to
# the same batch of log files.
# -----
10100|LogStatus||||1
10101|LogReport||||1
10102|LogUser||||1
10103|TmpStatus||||1
10104|TmpReport||||1
10105|TmpUser||||1
10110|MailFile||||1
#
# -----
# This parameter controls the Event Logger connection from the Toolkit.
# -----
10113|eventLogger.log||||1
#
# -----
# ASCII file which stores pointers to runtime SMF files in lieu of
# loading them to shared memory, which is a TK5 enhancement.
# -----
10111|ShmMem||||1
#

```

```

#
?   USER DEFINED RUNTIME PARAMETERS
#
#
# -----
# These parameters are required to support the PGS_SMF_Send...() tools.
# If the first parameter (TransmitFlag) is disabled, then none of the
# other parameters need to be set. By default, this functionality has been
# disabled. To enable, set TransmitFlag to 1 and supply the other 3
# parameters with local information.
# -----
10109|TransmitFlag; 1=transmit,0=disable|0
10106|RemoteHost|sandcrab
10107|RemotePath|/usr/kwan/test/PC/data
10108|EmailAddresses|kwan@eos.hitc.com
#
# -----
# This parameter controls the Event Logger connection from the Toolkit.
# -----
10112|Event Logging Flag; 1=connect,0=disconnect|1
#
# -----
# This entry defines the IP address of the processing host and is used
# by the Toolkit when generating unique Intermediate and Temporary file
# names. The Toolkit no longer relies on the PGS_HOST_PATH environment
# variable to obtain this information.
# -----
10099|Local IP Address of 'ether'|155.157.31.87
#

```

```

?   INTERMEDIATE INPUT

# Next line is default location for INTERMEDIATE INPUT FILES

!   ~/runtime
#
#

?   INTERMEDIATE OUTPUT

# Next line is default location for INTERMEDIATE OUTPUT FILES

!   ~/runtime
#
#

?   TEMPORARY I/O

# Next line is default location for TEMPORARY FILES

!   ~/runtime
#
#

?   END

```

C.2 Validating Process Control Files

C.2.1 DESCRIPTION:

The Process Control Information File Check Program is a program that checks the file containing the Process Control Status Information. This program is an aid to determine if the input file necessary for the Process Control Tools is in the proper format and contains the minimum amount of information for a valid run. The program is run by entering the program name followed by the file name to be checked. For example, "pccheck.sh -i userpcf.dat" will run the check program and check the file userpcf.dat located in the current directory. The -i flag needs to be followed by the name of the input file. Upon checking the file, a list of errors and warnings will be displayed to the user. Each error or warning will have a brief description, the line number, and the line itself. When the checking process has completed, a message appears stating that the check process is finished and the number of warnings and errors found are displayed. With this program, errors are defined as something in the file that, during execution of the Process Control Tools, the return will not be PGS_S_SUCCESS. A warning is defined as something that, although the Process Control Tools will return a PGS_S_SUCCESS, a problem could arise later. An example of this is the file name "file one.dat" is stored in the Process Control Information file. Upon execution, the Process Control Tools will return the name of this file and PGS_S_SUCCESS as the function type return value. When the program tries to open this file however, a file access error will occur.

C.2.2 INPUT

- Program name, -i flag, and file to be checked. An example of this would be:

```
pccheck.sh -i userpcf.dat
```

This will initiate the check program and check the file userpcf.dat in the current directory.

- Program name, -i flag, file to be checked, -o flag, and an output file name.

```
pccheck.sh -i userpcf.dat -o userpcf.out
```

This will initiate the check program and check the file userpcf.dat in the current directory and create an output file "outpct.fil" that will be an exact copy of userpcf.dat except the output file will contain line numbers.

- Program name, -h flag.

```
pccheck.sh -h
```

This will display a usage help message.

- Program name, -i flag, file to be checked, -c flag, and a template file name.

```
pccheck.sh -i userpcf.dat -c $PGSHOME/runtime/PCF.v3
```

This will list all errors and warnings in the file userpcf.dat and perform a comparison. The -c flag will initiate a comparison with a template file and determine if any of the Product ID's reserved by the PGS Toolkit (range 10,000 .. 10,999) differ in userpcf.dat and \$PGSHOME/runtime/PCF.v3. This will only list the differences and will not perform any corrections.

- Program name, -i flag, file to be checked, -c flag, and a template file name, -s flag, to suppress output.

```
pccheck.sh -i userpcf.dat -c $PGSHOME/runtime/PCF.v3 -s
```

The -s flag will suppress all output except that output received when using the -c flag. The -s flag is designed to be used only when the -c flag is used.

C.2.3 OUTPUT

List of errors and warnings followed by a summary of the number of errors and warnings. See the EXAMPLES section for detailed listings of program output. Using the -o flag will also allow the user to output a file that is an exact copy of the input file with line numbers in the file. This output option is provided as a convenience to the user; the output file is not intended to be used as the input Process Control Information File. Using the -c flag followed by a template file will allow the user to determine what reserved Logical Identifiers have been edited from the template file.

C.2.4 ERRORS:

The following is a list of possible errors followed by a brief description.

- "Unable to open input file: <file name>"—unable to open input file name passed in as a command line argument
- "Incorrect number of command line arguments"—the number of command line arguments did not match the number expected
- "Unexpectedly reached EOF"—the end of file was encountered before the correct number of dividers (?) were reached
- "Invalid number of system configuration parameters"—the number of system configuration parameters encountered did not match the number expected
- "Invalid index value in user defined configuration parameters"—an invalid index value was found
- "Problem with user defined configuration parameter"—user defined configuration parameter contains a problem (i.e., incorrect number of delimiters ()), or a value of all blanks)
- "Configuration value length too long"—user defined configuration value exceeds PGSD_PC_VALUE_LENGTH_MAX characters
- "Invalid index value involving file information"—an invalid index value was found in one of the sections that contains file information
- "Invalid number of delimiters involving file information"—line containing file information contains incorrect number of delimiters (())
- "No validity flag present in input file information"—validity flag is mandatory for input file information
- "File name length too long"—file name exceeds PGSD_PC_FILE_NAME_MAX characters
- "Path length too long"—path exceeds PGSD_PC_PATH_LENGTH_MAX characters
- "problem with version number in Standard input file information"—missing or unexpected sequence number
- "Default file location marker contains no data."
- "Default file location length too long."
- "Default file location not found."
- "Universal Reference length too long." - universal reference identifier exceeds PGSD_PC_UREF_LENGTH_MAX characters
- "File name does not exist." - File name data field is empty.

C.2.5 WARNINGS:

The following is a list of all possible warnings followed by a brief description.

- "Warning—Possible problem with system configuration value"—configuration parameter contains all blank characters
- "Warning—Repeat index number in user defined configuration parameters"—index value used twice in user defined configuration parameters
- "Warning—extra delimiters in user defined configuration parameters"—remaining delimiters will be returned as part of the value in user defined configuration parameters.
- "Warning—Repeat index number in file information"—index value illegally used multiple times in file information
- "Warning—possible problem in path or file name"—path or file name contains blank characters
- "Warning—information beyond final divider will be ignored"—anything after the last counted divider (?) will be ignored
- "Warning—possible problem in default file location."
- "Warning—Default file location not after divider."

C.2.6 EXAMPLES:

Three examples are provided below. Each example contains the input file used, the command entered and the corresponding output. The first example contains no errors or warnings. The second example contains several warnings and errors. The third example is an example of using the -c flag.

C.2.6.1 EXAMPLE 1

INPUT FILE: **userpcf.dat**

```
#
# Process Control File
#
#
? SYSTEM RUNTIME PARAMETERS
# -----
# Production Run ID - unique production instance identifier
# -----
```

```

1
# -----
# Software ID - unique software configuration identifier
# -----
1
#
?   PRODUCT INPUT FILES
# [ Default file location indicated by '!' ]
! ~/runtime
#
1000|temp.dat|/usr/atm/data||Optional Universal Reference|temp.att|1
1001|humid.dat|/usr/atm/data||Humidity Data|humid.att|1
600|wind_1.dat|||wind_1.att|2
600|wind_2.dat|||wind_2.att|1
# -----
# polar motion and UTC-UT1 file
# -----
10401|utcpole.dat|~/lib/database/CSC|||1
# -----
# earth model tags file
# -----
10402|earthfigure.dat|~/lib/database/CSC|||1
# -----
# JPL planetary ephemeris file (binary form)
# -----
10601|de200.eos|/usr/lib/database/CBP|||1
10964|fnocmin.imgswitched|||1
10965|fnocmod.imgswitched|||1
10966|srzarea.img|||1

```

```

10967|srzcode.img||||1
10968|srzphas.img||||1
10969|srzslop.img||||1
10970|srzsoil.img||||1
10971|srztext.img||||1
#
# -----
# The following are for the PGS_AA_dcw tool only.
# The IDs are #defined in the PGS_AA_dcw.h file
# -----
10990|eurnasia/||||1
10991|noamer/||||1
10992|soamafr/||||1
10993|sasaus/||||1
#
#
?   PRODUCT OUTPUT FILES
# [ Default file location indicated by '!' ]
! ~/runtime
#
1002|temp_lev3.hdf||||1
1003|humid_lev3.hdf||||1
601|wind_lev3.hdf||||1
#
#
?   SUPPORT INPUT FILES
# [ Default file location indicated by '!' ]
! ~/runtime
#

```

```

31|Wind_insitu.dat|/usr/wind/data|||1
#
#
# -----
# This ID is #defined in PGS_AA_Tools.h
# This file contains the IDs for all support and format files shown
# above
# -----
10900|indexFile|~/runtime|||1
#
# -----
# These are support files for the data set files - to be created by user
# (not necessarily a one-to-one relationship)
# The IDs must correspond to the logical IDs in the index file
# -----
10901|mowel13aSupport|~/runtime|||1
10902|owel13aSupport|~/runtime|||1
10903|owel14Support|~/runtime|||1
10904|etop05Support|~/runtime|||1
10905|fnoc1Support|~/runtime|||1
10906|fnoc2Support|~/runtime|||1
10907|zobler1Support|~/runtime|||1
10908|zobler2Support|~/runtime|||1
#
# -----
# The following are format files for each data set file
# (not necessarily a one-to-one relationship)
# The IDs must correspond to the logical IDs in the index file
# -----

```

```
10920|mowel13a.bfm|~/runtime||||1
10921|owel13a.bfm|~/runtime||||1
10922|owel14d.bfm|~/runtime||||1
10923|owel14dr.bfm|~/runtime||||1
10924|etop05.bfm|~/runtime||||1
10925|fnocAzm.bfm|~/runtime||||1
10926|fnocOcm.bfm|~/runtime||||1
10927|fnocPt.bfm|~/runtime||||1
10928|fnocRdg.bfm|~/runtime||||1
10929|fnocSt.bfm|~/runtime||||1
10930|fnocUrb.bfm|~/runtime||||1
10931|fnocWat.bfm|~/runtime||||1
10932|fnocMax.bfm|~/runtime||||1
10933|fnocMin.bfm|~/runtime||||1
10934|fnocMod.bfm|~/runtime||||1
10935|srzArea.bfm|~/runtime||||1
10936|srzCode.bfm|~/runtime||||1
10937|srzPhas.bfm|~/runtime||||1
10938|srzSlop.bfm|~/runtime||||1
10939|srzSoil.bfm|~/runtime||||1
10940|srzText.bfm|~/runtime||||1
#
#
?   SUPPORT OUTPUT FILES
# [ Default file location indicated by '!' ]
! ~/runtime
#
#
51|Wind_qlook.dat|/usr/wind/data||||1
```

```

#
# -----
# These files support the SMF log functionality. Each run will cause
# status information to be written to 1 or more of the Log files. To
# simulate DAAC operations, remove the 3 Logfiles between test runs.
# Remember: all executables within a PGE will contribute status data to
# the same batch of log files.
# -----
10100|LogStatus|~/runtime|||1
10101|LogReport|~/runtime|||1
10102|LogUser|~/runtime|||1
10103|TmpStatus|~/runtime|||1
10104|TmpReport|~/runtime|||1
10105|TmpUser|~/runtime|||1
10110|MailFile|~/runtime|||1
#
#
?   USER DEFINED RUNTIME PARAMETERS
3000|Humidity Instrument Calibration|0.34423772
3001|Temperature Instrument Calibration|1.87864
3002|Wind Instrument Calibration|0.992
3003|Atmospheric Algorithm|NIGHT
3004|Status Report Title|INSTRUMENT STATUS REPORT FOR LEVEL 2
#
#
# -----
# These parameters are required to support the PGS_SMF_Send...() tools.
# If the first parameter (TransmitFlag) is disabled, then none of the
# other parameters need to be set. By default, this functionality has

```

```
# been disabled. To enable, set TransmitFlag to 1 and supply the other 3
# parameters with local information.
# -----
10109|TransmitFlag; 1=transmit,0=disable|0
10106|RemoteHost|anyhost
10107|RemotePath|/usr/anyuser/anypath/data
10108|EmailAddresses|anyuser@anysystem.anyaddress.gov
#
#
?   INTERMEDIATE INPUT
# [ Default file location indicated by '!' ]
! ~/runtime
#
#
?   INTERMEDIATE OUTPUT
# [ Default file location indicated by '!' ]
! ~/runtime
#
#
?   TEMPORARY IO
# [ Default file location indicated by '!' ]
! ~/runtime
#
#
?   END
```

UNIX COMMAND LINE:

```
pccheck.sh -i userpcf.dat
```

```
Check of userpcf.dat completed
```

```
Errors found: 0
```

```
Warnings found: 0
```

C.2.6.2 EXAMPLE 2

INPUT FILE: userpcf.dat

```
#
# Process Control File
#
#
? SYSTEM RUNTIME PARAMETERS
# -----
# Production Run ID - unique production instance identifier
# -----
1
#***ONLY ONE SYSTEM CONFIGURATION PARAMETER***
#
? PRODUCT INPUT FILES
# [ Default file location marked by '!' ]
! ~/runtime
#
1000|temp.dat|/usr/atm/data||temp.att|
# ^ No version number****
1)01|humid.dat|/usr/atm/data||humid.att|1
#^Illegal character in index number****
600|wind_1.dat|||wind_1.att|2
600|wind_2.dat|||wind_2.att|1
# Line only contains five delimiters****
```

```

#
# -----
# polar motion and UTC-UT1 file
# -----
10401|utcpole.dat|~/lib/database/CSC||||1
# -----
# earth model tags file
# -----
10402|earthfigure.dat|~/lib/database/CSC||||1
# -----
# JPL planetary ephemeris file (binary form)
# -----
10601|de200.eos|/usr/lib/database/CBP||||1
10964|fnocmin.imgswitched||||1
10965|fnocmod.imgswitched||||1
10966|srzarea.img||||1
10967|srzcode.img||||1
10968|srzphas.img||||1
10969|srzslop.img||||1
10970|srzsoil.img||||1
10971|srztext.img||||1
#
# -----
# The following are for the PGS_AA_dcw tool only.
# The IDs are #defined in the PGS_AA_dcw.h file
# -----
10990|eurnesia/||||1
10991|noamer/||||1
10992|soamafr/||||1

```

```

10993|sasaus/|||||1
#
#
?   PRODUCT OUTPUT FILES
#
#   ^^^^ No default file location listed before first file name****
1002|temp_lev3.hdf|||||1
1003|humid_lev3.hdf|||||1
601|wind_lev3.hdf|||||1
#
#
?   SUPPORT INPUT FILES
# [ Default file location  marked by '!' ]
! ~/runtime
#
31|Wind_insitu .dat|/usr/wind/data|||||1
#           ^ Blank character in file name****
#
#
# -----
# This ID is #defined in PGS_AA_Tools.h
# This file contains the IDs for all support and format files shown
# above
# -----
10900|indexFile|~/runtime|||||1
#
# -----
# These are support files for the data set files - to be created by user
# (not necessarily a one-to-one relationship)

```

```

# The IDs must correspond to the logical IDs in the index file
# -----
10901|mowel13aSupport|~/runtime|||1
10902|owel13aSupport|~/runtime|||1
10903|owel14Support|~/runtime|||1
10904|etop05Support|~/runtime|||1
10905|fnoc1Support|~/runtime|||1
10906|fnoc2Support|~/runtime|||1
10907|zobler1Support|~/runtime|||1
10908|zobler2Support|~/runtime|||1
#
# -----
# The following are format files for each data set file
# (not necessarily a one-to-one relationship)
# The IDs must correspond to the logical IDs in the index file
# -----
10920|mowel13a.bfm|~/runtime|||1
10921|owel13a.bfm|~/runtime|||1
10922|owel14d.bfm|~/runtime|||1
10923|owel14dr.bfm|~/runtime|||1
10924|etop05.bfm|~/runtime|||1
10925|fnocAzm.bfm|~/runtime|||1
10926|fnocOcm.bfm|~/runtime|||1
10927|fnocPt.bfm|~/runtime|||1
10928|fnocRdg.bfm|~/runtime|||1
10929|fnocSt.bfm|~/runtime|||1
10930|fnocUrb.bfm|~/runtime|||1
10931|fnocWat.bfm|~/runtime|||1
10932|fnocMax.bfm|~/runtime|||1

```

```

10933|fnocMin.bfm|~/runtime||||1
10934|fnocMod.bfm|~/runtime||||1
10935|srzArea.bfm|~/runtime||||1
10936|srzCode.bfm|~/runtime||||1
10937|srzPhas.bfm|~/runtime||||1
10938|srzSlop.bfm|~/runtime||||1
10939|srzSoil.bfm|~/runtime||||1
10940|srzText.bfm|~/runtime||||1

#

#

?   SUPPORT OUTPUT FILES

# [ Default file location  marked by '!' ]

! ~/runtime

#

#

#

51|Wind_qlook.dat|/usr/wind/data||||1

#

# -----

# These files support the SMF log functionality. Each run will cause
# status information to be written to 1 or more of the Log files. To
# simulate DAAC operations, remove the 3 Logfiles between test runs.
# Remember: all executables within a PGE will contribute status data to
# the same batch of log files.

# -----

10100|LogStatus|~/runtime||||1
10101|LogReport|~/runtime||||1
10102|LogUser|~/runtime||||1
10103|TmpStatus|~/runtime||||1

```

```

10104|TmpReport|~/runtime|||1
10105|TmpUser|~/runtime|||1
10110|MailFile|~/runtime|||1
#
#
?   USER DEFINED RUNTIME PARAMETERS
3000|Humidity Instrument Calibration|0.34423772
3001|
#   ^ Incomplete line****
3002|Wind Instrument Calibration|0.992|
#                               ^ Extra delimiter****
3003|Atmospheric Algorithm|NIGHT
3001|Status Report Title|INSTRUMENT STATUS REPORT FOR LEVEL 2
#   Index number used six lines above****
#
#
# -----
# These parameters are required to support the PGS_SMF_Send...() tools.
# If the first parameter (TransmitFlag) is disabled, then none of the
# other parameters need to be set. By default, this functionality has
# been disabled. To enable, set TransmitFlag to 1 and supply the other 3
# parameters with local information.
# -----
10109|TransmitFlag; 1=transmit,0=disable|0
10106|RemoteHost|anyhost
10107|RemotePath|/usr/anyuser/anypath/data
10108|EmailAddresses|anyuser@anysystem.anyaddress.gov
#
#

```

```
? INTERMEDIATE INPUT
# [ Default file location marked by '!' ]
! ~/runtime
#
#
#
? INTERMEDIATE OUTPUT
# [ Default file location marked by '!' ]
! ~/runtime
#
#
#
? TEMPORARY IO
# [ Default file location marked by '!' ]
! ~/runtime
#
#
#
? END
# We just passed the last divider****
```

UNIX COMMAND LINE:-

```
pccheck.sh -i userpcf.dat -o userpcf.out
```

```
Error - Invalid number of system configuration parameters.
```

```
Found: 1
```

```
Expected: 2
```

```
Error - problem with version number in Standard input or output file
information.
```

```
Line number: 16
```

```
Line: 1000|temp.dat|/usr/atm/data||temp.att|
```

Error - Invalid identifier number involving file information.

Line number: 18

Line: 1)01|humid.dat|/usr/atm/data|||humid.att|1

Error - Invalid number of delimiters involving file information.

Line number: 21

Line: 600|wind_2.dat|||wind_2.att|1

Error - Default file location not found.

Line number: 58

Line: 1002|temp_lev3.hdf|||||1

Warning - possible problem in path or file name.

Line number: 67

Line: 31|Wind_insitu .dat|/usr/wind/data|||||1

Error - Problem with user defined configuration parameter.

Line number: 146

Line: 3001|

Warning - extra delimiters in user defined configuration parameters.

Line number: 148

Line: 3002|Wind Instrument Calibration|0.992|

Warning - Repeat index number in user defined configuration parameters.

Line number: 151

Line: 3001|Status Report Title|INSTRUMENT STATUS REPORT FOR LEVEL 2

Warning - information beyond final divider will be ignored.

line number: 185

Number of dividers read: 10

Number of dividers expected: 10

Check of usrpcf.dat completed

Errors found: 6

Warnings found: 4

OUTPUT FILE: usrpcf.out

```
1:#
2:# Process Control File
3:#
4:#
5:? SYSTEM RUNTIME PARAMETERS
6:# -----
7 :# Production Run ID - unique production instance identifier
8:# -----
9:1
10:#****ONLY ONE SYSTEM CONFIGURATION PARAMETER****
11:#
12:? PRODUCT INPUT FILES
13:# [ Default file location marked by '!' ]
14:! ~/runtime
15:#
16:1000|temp.dat|/usr/atm/data||temp.att|
17:# ^ No version number****
18:1)01|humid.dat|/usr/atm/data||humid.att|1
```

```

19:#^Illegal character in index number****
20:600|wind_1.dat|||wind_1.att|2
21:600|wind_2.dat|||wind_2.att|1
22:# Line only contains five delimiters****
23:#
24:# -----
25:# polar motion and UTC-UT1 file
26:# -----
27:10401|utcpole.dat|~/lib/database/CSC|||1
28:# -----
29:# earth model tags file
30:# -----
31:10402|earthfigure.dat|~/lib/database/CSC|||1
32:# -----
33:# JPL planetary ephemeris file (binary form)
34:# -----
35:10601|de200.eos|/usr/lib/database/CBP|||1
36:10964|fnocmin.imgswitched|||1
37:10965|fnocmod.imgswitched|||1
38:10966|srzarea.img|||1
39:10967|srzcode.img|||1
40:10968|srzphas.img|||1
41:10969|srzslop.img|||1
42:10970|srzsoil.img|||1
43:10971|srztext.img|||1
44:#
45:# -----
46:# The following are for the PGS_AA_dcw tool only.
47:# The IDs are #defined in the PGS_AA_dcw.h file

```

```

48:# -----
49:10990|eurnesia/||||1
50:10991|noamer/||||1
51:10992|soamafrr/||||1
52:10993|sasaus/||||1
53:#
54:#
55:?    PRODUCT OUTPUT FILES
56:#
57:# ^^^^ No default file location listed before first file name****
58:1002|temp_lev3.hdf||||1
59:1003|humid_lev3.hdf||||1
60:601|wind_lev3.hdf||||1
61:#
62:#
63:?    SUPPORT INPUT FILES
64:# [ Default file location  marked by '!' ]
65:! ~/runtime
66:#
67:31|Wind_insitu .dat|/usr/wind/data||||1
68:#          ^ Blank character in file name****
69:#
70:#
71:# -----
72:# This ID is #defined in PGS_AA_Tools.h
73:# This file contains the IDs for all support and format files shown
74:# above
75:# -----
76:10900|indexFile|~/runtime||||1

```

```

77:#
78:# -----
79:# These are support files for the data set files - to be created by user
80:# (not necessarily a one-to-one relationship)
81:# The IDs must correspond to the logical IDs in the index file
82:# -----
83:10901|mowel3aSupport|~/runtime|||1
84:10902|owel3aSupport|~/runtime|||1
85:10903|owel4Support|~/runtime|||1
86:10904|etop05Support|~/runtime|||1
87:10905|fnoc1Support|~/runtime|||1
88:10906|fnoc2Support|~/runtime|||1
89:10907|zobler1Support|~/runtime|||1
90:10908|zobler2Support|~/runtime|||1
91:#
92:# -----
93:# The following are format files for each data set file
94:# (not necessarily a one-to-one relationship)
95:# The IDs must correspond to the logical IDs in the index file
96:# -----
97:10920|mowel3a.bfm|~/runtime|||1
98:10921|owel3a.bfm|~/runtime|||1
99:10922|owel4d.bfm|~/runtime|||1
100:10923|owel4dr.bfm|~/runtime|||1
101:10924|etop05.bfm|~/runtime|||1
102:10925|fnocAzm.bfm|~/runtime|||1
103:10926|fnocOcm.bfm|~/runtime|||1
104:10927|fnocPt.bfm|~/runtime|||1
105:10928|fnocRdg.bfm|~/runtime|||1

```

```

106:10929|fnocSt.bfm|~/runtime||||1
107:10930|fnocUrb.bfm|~/runtime||||1
108:10931|fnocWat.bfm|~/runtime||||1
109:10932|fnocMax.bfm|~/runtime||||1
110:10933|fnocMin.bfm|~/runtime||||1
111:10934|fnocMod.bfm|~/runtime||||1
112:10935|srzArea.bfm|~/runtime||||1
113:10936|srzCode.bfm|~/runtime||||1
114:10937|srzPhas.bfm|~/runtime||||1
115:10938|srzSlop.bfm|~/runtime||||1
116:10939|srzSoil.bfm|~/runtime||||1
117:10940|srzText.bfm|~/runtime||||1
118:#
119:#
120:??   SUPPORT OUTPUT FILES
121:# [ Default file location marked by '!' ]
122:! ~/runtime
123:#
124:#
125:#
126:51|Wind_qlook.dat|/usr/wind/data||||1
127:#
128:# -----
129:# These files support the SMF log functionality. Each run will cause
130:# status information to be written to 1 or more of the Log files. To
131:# simulate DAAC operations, remove the 3 Logfiles between test runs.
132:# Remember: all executables within a PGE will contribute status data to
133:# the same batch of log files.
134:# -----

```

```

135:10100|LogStatus|~/runtime|||1
136:10101|LogReport|~/runtime|||1
137:10102|LogUser|~/runtime|||1
138:10103|TmpStatus|~/runtime|||1
139:10104|TmpReport|~/runtime|||1
140:10105|TmpUser|~/runtime|||1
141:10110|MailFile|~/runtime|||1
142:#
143:#
144:?    USER DEFINED RUNTIME PARAMETERS
145:3000|Humidity Instrument Calibration|0.34423772
146:3001|
147:#    ^ Incomplete line****
148:3002|Wind Instrument Calibration|0.992|
149:#                                ^ Extra delimiter****
150:3003|Atmospheric Algorithm|NIGHT
151:3001|Status Report Title|INSTRUMENT STATUS REPORT FOR LEVEL 2
152:#    Index number used six lines above****
153:#
154:#
155:# -----
156:# These parameters are required to support the PGS_SMF_Send...() tools.
157:# If the first parameter (TransmitFlag) is disabled, then none of the
158:# other parameters need to be set. By default, this functionality has
159:# been disabled. To enable, set TransmitFlag to 1 and supply the other 3
160:# parameters with local information.
161:# -----
162:10109|TransmitFlag; 1=transmit,0=disable|0
163:10106|RemoteHost|anyhost

```

```
164:10107|RemotePath|/usr/anyuser/anypath/data
165:10108|EmailAddresses|anyuser@anysystem.anyaddress.gov
166:#
167:#
168:?   INTERMEDIATE INPUT
169:# [ Default file location  marked by '!' ]
170:! ~/runtime
171:#
172:#
173:#
174:?   INTERMEDIATE OUTPUT
175:# [ Default file location  marked by '!' ]
176:! ~/runtime
177:#
178:#
179:?   TEMPORARY IO
180:# [ Default file location  marked by '!' ]
181:! ~/runtime
182:#
183:#
184:?   END
185:#           We just passed the last divider*****
```

C.2.6.3 EXAMPLE 3

INPUT FILE: userpcf.dat

```
#
#   Process Control File
#
#
```

```

?   SYSTEM RUNTIME PARAMETERS

# -----

# Production Run ID - unique production instance identifier

# -----

1

# -----

# Software ID - unique software configuration identifier

# -----

1

#

?   PRODUCT INPUT FILES

# [ Default file location marked by '!' ]

! ~/runtime

# -----

# These are actual ancillary data set files - supplied by ECS or the user
# the following are supplied for purposes of tests and as a useful set of
# ancillary data.

# -----

10780|usatile12|||10751|12
10780|usatile11|||10750|11
10780|usatile10|||10749|10
10780|usatile9|||10748|9
10780|usatile8|||10747|8
10780|usatile7|||10746|7
10780|usatile6|||10745|6
10780|usatile5|||10744|5
10780|usatile4|||10743|4
10780|usatile3|||10742|3
10780|usatile2|||10741|2

```

10780|usatile1||||10740|1
10951|mowel3a.img||||1
10952|owel3a.img||||1
10953|owel14d.img||||1
10954|owel14dr.img||||1
10955|etop05.dat||||1
10956|fnocazm.img||||1
10957|fnococm.img||||1
10958|fnocpt.img||||1
10959|fnocrdg.img||||1
10960|fnocst.img||||1
10961|fnocurb.img||||1
10962|fnocwat.img||||1
10963|fnocmax.imgs||||1
10964|fnocmin.imgs||||1
10965|fnocmod.imgs||||1
10966|srzarea.img||||1
10967|srzcode.img||||1
10968|srzphas.img||||1
10969|srzslop.img||||1
10970|srzsoil.img||||1
10971|srztext.img||||1
10972|nmcRucPotPres.datrepack||||1
10973|tbase.bin||||10915|1
10974|tbase.br||||10919|4
10974|tbase.bl||||10918|3
10974|tbase.tr||||10917|2
10974|tbase.tl||||10916|1
10975|geoid.dat||||1

```

#
# -----
# The following are for the PGS_GCT tool only.
# The IDs are #defined in the PGS_GCT.h file
# -----
10200|nad27sp|~/runtime|||1
10201|nad83sp|~/runtime|||1
# -----
# The following are for the PGS_AA_DCW tool only.
# The IDs are #defined in the PGS_AA_DCW.h file
# -----
10990|eurnesia/||||1
10991|noamer/||||1
10992|soamafr/||||1
10993|sasaus/||||1
#
1000|temp.dat|/usr/atm/data||temp.att|1
1001|humid.dat|/usr/atm/data||humid.att|1
600|wind_1.dat|||wind_1.att|2
600|wind_2.dat|||wind_2.att|1
# -----
# polar motion and UTC-UT1 file
# -----
10401|utcpole.dat|~/lib/database/CSC||||1
# -----
# earth model tags file
# -----
10402|earthfigure.dat|~/lib/database/CSC||||1
# -----

```

```

# JPL planetary ephemeris file (binary form)
# -----
10601|de200.eos|/usr/lib/database/CBP||||1
#
#
?   PRODUCT OUTPUT FILES
# [ Default file location marked by '!' ]
! ~/runtime
#
1002|temp_lev3.hdf||||1
1003|humid_lev3.hdf||||1
601|wind_lev3.hdf||||1
#
#
?   SUPPORT INPUT FILES
# [ Default file location marked by '!' ]
! ~/runtime
#
31|Wind_insitu.dat|/usr/wind/data||||1
#
#
# -----
# This ID is #defined in PGS_AA_Tools.h
# This file contains the IDs for all support and format files shown
# above
# -----
10900|indexFile|~/runtime||||1
#
# -----

```

```

# These are support files for the data set files - to be created by user
# (not necessarily a one-to-one relationship)
# The IDs must correspond to the logical IDs in the index file
# -----
10901|mowel13aSupport|~/runtime|||1
10902|owel13aSupport|~/runtime|||1
10903|owel14Support|~/runtime|||1
10904|etop05Support|~/runtime|||1
10905|fnoc1Support|~/runtime|||1
10906|fnoc2Support|~/runtime|||1
10907|zobler1Support|~/runtime|||1
10908|zobler2Support|~/runtime|||1
10909|nmcRucSupport|~/runtime|||1
10915|tbaseSupport|~/runtime|||1
10916|tbase1Support|~/runtime|||1
10917|tbase2Support|~/runtime|||1
10918|tbase3Support|~/runtime|||1
10919|tbase4Support|~/runtime|||1
10740|usatile1Support|~/runtime|||1
10741|usatile2Support|~/runtime|||1
10742|usatile3Support|~/runtime|||1
10743|usatile4Support|~/runtime|||1
10744|usatile5Support|~/runtime|||1
10745|usatile6Support|~/runtime|||1
10746|usatile7Support|~/runtime|||1
10747|usatile8Support|~/runtime|||1
10748|usatile9Support|~/runtime|||1
10749|usatile10Support|~/runtime|||1
10750|usatile11Support|~/runtime|||1

```

```

10751|usatile12Support|~/runtime|||1
10948|geoidSupport|~/runtime|||1
#
# -----
# The following are format files for each data set file
# (not necessarily a one-to-one relationship)
# The IDs must correspond to the logical IDs in the index file
# -----
10920|mowel13a.bfm|~/runtime|||1
10921|owel13a.bfm|~/runtime|||1
10922|owel14d.bfm|~/runtime|||1
10923|owel14dr.bfm|~/runtime|||1
10924|etop05.bfm|~/runtime|||1
10925|fnocAzm.bfm|~/runtime|||1
10926|fnocOcm.bfm|~/runtime|||1
10927|fnocPt.bfm|~/runtime|||1
10928|fnocRdg.bfm|~/runtime|||1
10929|fnocSt.bfm|~/runtime|||1
10930|fnocUrb.bfm|~/runtime|||1
10931|fnocWat.bfm|~/runtime|||1
10932|fnocMax.bfm|~/runtime|||1
10933|fnocMin.bfm|~/runtime|||1
10934|fnocMod.bfm|~/runtime|||1
10935|srzArea.bfm|~/runtime|||1
10936|srzCode.bfm|~/runtime|||1
10937|srzPhas.bfm|~/runtime|||1
10938|srzSlop.bfm|~/runtime|||1
10939|srzSoil.bfm|~/runtime|||1
10940|srzText.bfm|~/runtime|||1

```

```

#
#
?   SUPPORT OUTPUT FILES
# [ Default file location marked by '!' ]
! ~/runtime
#
#
51|Wind_qlook.dat|/usr/wind/data|||1
#
# -----
# These files support the SMF log functionality. Each run will cause
# status information to be written to 1 or more of the Log files. To
# simulate DAAC operations, remove the 3 Logfiles between test runs.
# Remember: all executables within a PGE will contribute status data to
# the same batch of log files.
# -----
10100|LogStatus|~/runtime|||1
10101|LogReport|~/runtime|||1
10102|LogUser|~/runtime|||1
10103|TmpStatus|~/runtime|||1
10104|TmpReport|~/runtime|||1
10105|TmpUser|~/runtime|||1
10110|MailFile|~/runtime|||1
#
#
?   USER DEFINED RUNTIME PARAMETERS
3000|Humidity Instrument Calibration|0.34423772
3001|Temperature Instrument Calibration|1.87864
3002|Wind Instrument Calibration|0.992

```

```

3003|Atmospheric Algorithm|NIGHT
3004|Status Report Title|INSTRUMENT STATUS REPORT FOR LEVEL 2
#
#
# -----
# These parameters are required to support the PGS_SMF_Send...() tools.
# If the first parameter (TransmitFlag) is disabled, then none of the
# other parameters need to be set. By default, this functionality has
# been disabled. To enable, set TransmitFlag to 1 and supply the other 3
# parameters with local information.
# -----
10109|TransmitFlag; 1=transmit,0=disable|0
10106|RemoteHost|anyhost
10107|RemotePath|/usr/anyuser/anypath/data
10108|EmailAddresses|anyuser@anysystem.anyaddress.gov
#
#
?   INTERMEDIATE INPUT
# [ Default file location marked by '!' ]
! ~/runtime
#
#
?   INTERMEDIATE OUTPUT
# [ Default file location marked by '!' ]
! ~/runtime
#
#
?   TEMPORARY IO
# [ Default file location marked by '!' ]

```

```
! ~/runtime
```

```
#
```

```
#
```

```
? END
```

COMPARISON FILE: PCF.testmaster

```
#
```

```
# filename:
```

```
#     PCF.testmaster
```

```
#
```

```
# description:
```

```
#     Process Control File (PCF)
```

```
#
```

```
# notes:
```

```
#
```

```
#     This file supports the IR-1 version of the toolkit.
```

```
#
```

```
#     Please treat this file as a master template and make copies of it  
#     for your own testing. Note that the Toolkit installation script sets  
#     PGS_PC_INFO_FILE to point to this master file by default. Remember  
#     to reset the environment variable PGS_PC_INFO_FILE to point to the  
#     instance of your PCF.
```

```
#
```

```
# -----
```

```
?     SYSTEM RUNTIME PARAMETERS
```

```
# -----
```

```
# Production Run ID - unique production instance identifier
```

```
# -----
```

```
1
```

```

# -----
# Software ID - unique software configuration identifier
# -----
1
#
?   PRODUCT INPUT FILES
# Next non-comment line is the default location for PRODUCT INPUT FILES
# WARNING! DO NOT MODIFY THIS LINE unless you have relocated these
# data set files to the location specified by the new setting.
!   ~/runtime
#
# -----
# These are actual ancillary data set files - supplied by ECS or the user
# the following are supplied for purposes of tests and as a useful set of
# ancillary data.
# -----
10780|usatile12|||10751|12
10780|usatile11|||10750|11
10780|usatile10|||10749|10
10780|usatile9|||10748|9
10780|usatile8|||10747|8
10780|usatile7|||10746|7
10780|usatile6|||10745|6
10780|usatile5|||10744|5
10780|usatile4|||10743|4
10780|usatile3|||10742|3
10780|usatile2|||10741|2
10780|usatile1|||10740|1
10951|mowel3a.img|||1

```

10952|owe13a.img||||1
10953|owe14d.img||||1
10954|owe14dr.img||||1
10955|etop05.dat||||1
10956|fnocazm.img||||1
10957|fnococm.img||||1
10958|fnocpt.img||||1
10959|fnocrdg.img||||1
10960|fnocst.img||||1
10961|fnocurb.img||||1
10962|fnocwat.img||||1
10963|fnocmax.imgs||||1
10964|fnocmin.imgs||||1
10965|fnocmod.imgs||||1
10966|srzarea.img||||1
10967|srzcode.img||||1
10968|srzphas.img||||1
10969|srzslop.img||||1
10970|srzsoil.img||||1
10971|srztext.img||||1
10972|nmcRucPotPres.datrepack||||1
10973|tbase.bin||||10915|1
10974|tbase.br||||10919|4
10974|tbase.bl||||10918|3
10974|tbase.tr||||10917|2
10974|tbase.tl||||10916|1
10975|geoid.dat||||1

```

# The following are for the PGS_GCT tool only.
# The IDs are #defined in the PGS_GCT.h file
# -----
10200|nad27sp|~/runtime||||1
10201|nad83sp|~/runtime||||1
# -----
# The following are for the PGS_AA_DCW tool only.
# The IDs are #defined in the PGS_AA_DCW.h file
# -----
10990|eurnesia/||||1
10991|noamer/||||1
10992|soamafr/||||1
10993|sasaus/||||1
#
#
?   PRODUCT OUTPUT FILES
# Next line is the default location for PRODUCT OUTPUT FILES
!   ~/runtime
#
#
?   SUPPORT INPUT FILES
# Next line is the default location for SUPPORT INPUT FILES
!   ~/runtime
#
#
# -----
# This ID is #defined in PGS_AA_Tools.h
# This file contains the IDs for all support and format files shown above
# -----

```

```

10900|indexFile|~/runtime||||1
#
# -----
# These are support files for the data set files - to be created by user
# (not necessar
# The IDs must correspond to the logical IDs in the index file
# -----
10901|mowel13aSupport|~/runtime||||1
10902|owel13aSupport|~/runtime||||1
10903|owel14Support|~/runtime||||1
10904|etop05Support|~/runtime||||1
10905|fnoc1Support|~/runtime||||1
10906|fnoc2Support|~/runtime||||1
10907|zobler1Support|~/runtime||||1
10908|zobler2Support|~/runtime||||1
10909|nmcRucSupport|~/runtime||||1
10915|tbaseSupport|~/runtime||||1
10916|tbase1Support|~/runtime||||1
10917|tbase2Support|~/runtime||||1
10918|tbase3Support|~/runtime||||1
10919|tbase4Support|~/runtime||||1
10740|usatile1Support|~/runtime||||1
10741|usatile2Support|~/runtime||||1
10742|usatile3Support|~/runtime||||1
10743|usatile4Support|~/runtime||||1
10744|usatile5Support|~/runtime||||1
10745|usatile6Support|~/runtime||||1
10746|usatile7Support|~/runtime||||1
10747|usatile8Support|~/runtime||||1

```

```

10748|usatile9Support|~/runtime|||1
10749|usatile10Support|~/runtime|||1
10750|usatile11Support|~/runtime|||1
10751|usatile12Support|~/runtime|||1
10948|geoidSupport|~/runtime|||1
#
# -----
# The following are format files for each data set file
# (not necessarily a one-to-one relationship)
# The IDs must correspond to the logical IDs in the index file
# -----
10920|mowel13a.bfm|~/runtime|||1
10921|owel13a.bfm|~/runtime|||1
10922|owel14d.bfm|~/runtime|||1
10923|owel14dr.bfm|~/runtime|||1
10924|etop05.bfm|~/runtime|||1
10925|fnocAzm.bfm|~/runtime|||1
10926|fnoc0cm.bfm|~/runtime|||1
10927|fnocPt.bfm|~/runtime|||1
10928|fnocRdg.bfm|~/runtime|||1
10929|fnocSt.bfm|~/runtime|||1
10930|fnocUrb.bfm|~/runtime|||1
10931|fnocWat.bfm|~/runtime|||1
10932|fnocMax.bfm|~/runtime|||1
10933|fnocMin.bfm|~/runtime|||1
10934|fnocMod.bfm|~/runtime|||1
10935|srzArea.bfm|~/runtime|||1
10936|srzCode.bfm|~/runtime|||1
10937|srzPhas.bfm|~/runtime|||1

```

```

10938|srzSlop.bfm|~/runtime|||1
10939|srzSoil.bfm|~/runtime|||1
10940|srzText.bfm|~/runtime|||1
10704|usatile5.bfm|~/runtime|||1
10705|usatile6.bfm|~/runtime|||1
10706|usatile7.bfm|~/runtime|||1
10707|usatile8.bfm|~/runtime|||1
10708|usatile9.bfm|~/runtime|||1
10709|usatile10.bfm|~/runtime|||1
10710|usatile11.bfm|~/runtime|||1
10711|usatile12.bfm|~/runtime|||1
10947|geoidbfm|~/runtime|||1
#
#
# -----
# leap seconds (TAI-UTC) file
# -----
10301|leapsec.dat|~/database/sun5/TD|||1
#
# -----
# polar motion and UTC-UT1 file
# -----
10401|utcpole.dat|~/database/sun5/CSC|||1
#
# -----
# earth model tags file
# -----
10402|earthfigure.dat|~/database/sun5/CSC|||1
#

```

```

# -----
# directory where spacecraft ephemeris files are located
# NOTE: This line is used to specify a directory only!
#     The "file" field should not be altered.
# -----
10501|. |~/database/sun5/EPH|||1
#
# -----
# JPL planetary ephemeris file (binary form)
# -----
10601|de200.eos|~/database/sun5/CBP|||1
#
#
?   SUPPORT OUTPUT FILES
# Next line is default location for SUPPORT OUTPUT FILES
!   ~/runtime
#
#
# -----
# These files support the SMF log functionality. Each run will cause
# status information to be written to 1 or more of the Log files. To
# simulate DAAC operations, remove the 3 Logfiles between test runs.
# Remember: all executables within a PGE will contribute status data to
# the same batch of log files.
# -----
10100|LogStatus|||1
10101|LogReport|||1
10102|LogUser|||1
10103|TmpStatus|||1

```

```

10104|TmpReport||||1
10105|TmpUser||||1
10110|MailFile||||1
#
# -----
# This parameter controls the Event Logger connection from the Toolkit.
# -----
10113|eventLogger.log||||1
#
# -----
# ASCII file which stores pointers to runtime SMF files in lieu of
# loading them to shared memory, which is a TK5 enhancement.
# -----
10111|ShmMem||||1
#
#
?   USER DEFINED RUNTIME PARAMETERS
#
#
# -----
# These parameters are required to support the PGS_SMF_Send...() tools.
# If the first parameter (TransmitFlag) is disabled, then none of the
# other parameters need to be set. By default, this functionality has been
# disabled. To enable, set TransmitFlag to 1 and supply the other 3
# parameters with local information.
# -----
10109|TransmitFlag; 1=transmit,0=disable|0
10106|RemoteHost|sandcrab
10107|RemotePath|/usr/kwan/test/PC/data

```

```

10108|EmailAddresses|kwan@eos.hitc.com
#
# -----
# This parameter controls the Event Logger connection from the Toolkit.
# -----
10112|Event Logging Flag; 1=connect,0=disconnect|1
#
# -----
# This entry defines the IP address of the processing host and is used
# by the Toolkit when generating unique Intermediate and Temporary file
# names. The Toolkit no longer relies on the PGS_HOST_PATH environment
# variable to obtain this information.
# -----
10099|Local IP Address of 'ether'|155.157.31.87
#
?   INTERMEDIATE INPUT
# Next line is default location for INTERMEDIATE INPUT FILES
!   ~/runtime
#
#
?   INTERMEDIATE OUTPUT
# Next line is default location for INTERMEDIATE OUTPUT FILES
!   ~/runtime
#
#
?   TEMPORARY I/O
# Next line is default location for TEMPORARY FILES
!   ~/runtime
#

```

```
#  
?   END
```

UNIX COMMAND LINE:

```
pccheck.sh -i userpcf.dat -c PCF.testmaster -s
```

The following lines were listed in the template file: PCF.testmaster
and have been altered or deleted from the input file.

```
> 10704|usatile5.bfm|~/runtime|||1  
> 10705|usatile6.bfm|~/runtime|||1  
> 10706|usatile7.bfm|~/runtime|||1  
> 10707|usatile8.bfm|~/runtime|||1  
> 10708|usatile9.bfm|~/runtime|||1  
> 10709|usatile10.bfm|~/runtime|||1  
> 10710|usatile11.bfm|~/runtime|||1  
> 10711|usatile12.bfm|~/runtime|||1  
> 10947|geoidbfm|~/runtime|||1  
> 10301|leapsec.dat|~/database/sun5/TD|||1  
> 10401|utcpole.dat|~/database/sun5/CSC|||1  
> 10402|earthfigure.dat|~/database/sun5/CSC|||1  
> 10501|. |~/database/sun5/EPH|||1  
> 10601|de200.eos|~/database/sun5/CBP|||1  
> 10100|LogStatus|||1  
> 10101|LogReport|||1  
> 10102|LogUser|||1  
> 10103|TmpStatus|||1  
> 10104|TmpReport|||1  
> 10105|TmpUser|||1  
> 10110|MailFile|||1
```

```

> 10113|eventLogger.log||||1
> 10111|ShmMem||||1
> 10106|RemoteHost|sandcrab
> 10107|RemotePath|/usr/kwan/test/PC/data
> 10108|EmailAddresses|kwan@eos.hitc.com
> 10112|Event Logging Flag; 1=connect,0=disconnect|1
> 10099|Local IP Address of 'ether'|155.157.31.87

```

These are the lines in the input file: usrpcf.dat
that differ from the template file.

```

< 10401|utcpole.dat|~/lib/database/CSC||||1
< 10402|earthfigure.dat|~/lib/database/CSC||||1
< 10601|de200.eos|/usr/lib/database/CBP||||1
< 10100|LogStatus|~/runtime||||1
< 10101|LogReport|~/runtime||||1
< 10102|LogUser|~/runtime||||1
< 10103|TmpStatus|~/runtime||||1
< 10104|TmpReport|~/runtime||||1
< 10105|TmpUser|~/runtime||||1
< 10110|MailFile|~/runtime||||1
< 10106|RemoteHost|anyhost
< 10107|RemotePath|/usr/anyuser/anypath/data
< 10108|EmailAddresses|anyuser@anysystem.anyaddress.gov

```

C.2.7 BENEFITS:

Due to the fact that the Process Control Information file must currently be entered by hand, errors can easily be introduced. Many errors are not obvious and may not be detected by the Process Control Tools. By adopting the practice of using this utility to check your PCF after each modification, the number of runtime errors can be greatly reduced.

Appendix D. Ancillary Data Access Tools

This appendix deals with the use of the ancillary data access tools:

PGS_AA_dcw

PGS_AA_dem

PGS_AA_2DRead

PGS_AA_2Dgeo

PGS_AA_3DRead

PGS_AA_3Dgeo

PGS_AA_PeVA

The first section below describes how the tools are conceived. Each tool is then described in terms of

- the data set(s) to which it is designed to give access including its accuracy and precision
- an outline of the means by which the tool achieves access and any options available through the calling sequence.
- how the user can call the tool to optimize resource efficiency
- upgrade possibilities

The DCW tool is described in the second section; while the DEM, 2 and 3 D tools, being closely allied in functional terms are described together in the third section. The fourth section describes the Parameter = Value tool that is a support tool for the other tools but can also be used directly by science users in algorithms.

This information is additional to that in the main User Guide pages and calling sequence details are not repeated here.

D.1 Introduction

The ancillary data tools are optional for use in science algorithms. There is a wide range of ancillary data sets and these tools have been designed to provide useful access functionality only for those data sets for which generic functionality can be provided centrally.

Users could utilize language standard input/output functions or the HDF tools to access the ancillary data. However, a suite of higher level tools is required for four reasons:

- a to enable data from locations specified by the user to be returned to the user thus avoiding having to know the internal structure of the file.

- b to shield the user from having to know details of parameter source or source format or to track changes in either, although sources changes will be agreed with the user.
- c to provide for certain additional manipulations of extracted data.

For this final point (c), only those data sets that have been specifically identified as requiring particular manipulations will be serviced; i.e., the ancillary tools do not intend to provide a general manipulation service for all types of data. However, the tools which 'extract from location' (a) will be sufficiently generic to allow additional data sets of a similar type to be used.

Access to the information will be in response to an algorithm request in the form of pointers to parameters and locations in a data file. These pointers take the form of a latitude and longitude or a similar two dimensional or three dimensional pointer.

It has been assumed that users require to access single or multiple point locations for one or more parameters and that these values will be returned in arrays to the user. This is in sharp contrast to the other major use of various ancillary data that are used for display purposes on screens. It is further assumed that the user requires multiple extractions made in user defined loops; very often driven by the systematic examination of time ordered source packets along or orthogonal to the sub-satellite track.

D.2 PGS_AA_dcw

D.2.1 Data Sets Accessed

PGS_AA_dcw is an ancillary data tool to be used to access the Digital Chart of the World database (DCW). The tool can only be used for accessing DCW.

A subset of the DCW database subset is delivered with the tool. For descriptions of data sets and file structure see **Digital Chart of the World—Final DCW Product Specification MIL-D-89009 December 7, 1991**.

DCW is a general purpose digital global database designed for Geographical Information Systems (GIS) applications. It utilizes a vector based, thematically layered data set available on four CD-ROM's at a comprehensive scale of 1:1,000,000. It consists of geographic, attribute, and textual data, stored in Vector Product Format or VPF. VPF is described in **Vector Product Format (MIL-STD-60006)**.

The data provided with the tool is exactly as found in the product, therefore any errors are a result of the database and not the tool. The DCW content is based primarily on the feature content of the 1:1,000,000-scale DMA Operational Navigational Chart (ONC) series. The 270 ONC sheets are supplemented with six 1:2,000,000-scale Jet Navigation Charts (JNC's) in the Antarctic region where ONC coverage is not available.

The absolute horizontal accuracy of the DCW for all features derived from ONC's is < 2040 meters (<6700 feet) rounded to the nearest 5 meters at 90% Circular Error (CE), World Geodetic System (PGSD_WGS84). The absolute horizontal accuracy for all features derived from JNC's is <4270 meters (<14000 feet) at 90% CE.

DCW is provided normally on four CDROM'S comprising of more than 1500MB of data. Requirements from PGS_AA_dcw were to provide land/sea/ice flags for the world, so the relevant coverage from the data base was extracted; namely Political/Oceans. This coverage contains all the vector information pertaining to political boundaries and those which exist between certain cover types i.e., land/sea/ice. (DCW states that the representation of international boundaries is not authoritative)

The structure of the DCW database is represented in Fig 1. The DCW database implements three types of VPF files: directories, tables and indices. The data base files are contained within a hierarchy of directories. Contained within these directories are the tables and indices that provide information. Each table within the database consists of two parts the header and the data records. By examining the header, it is possible to locate the information wanted.

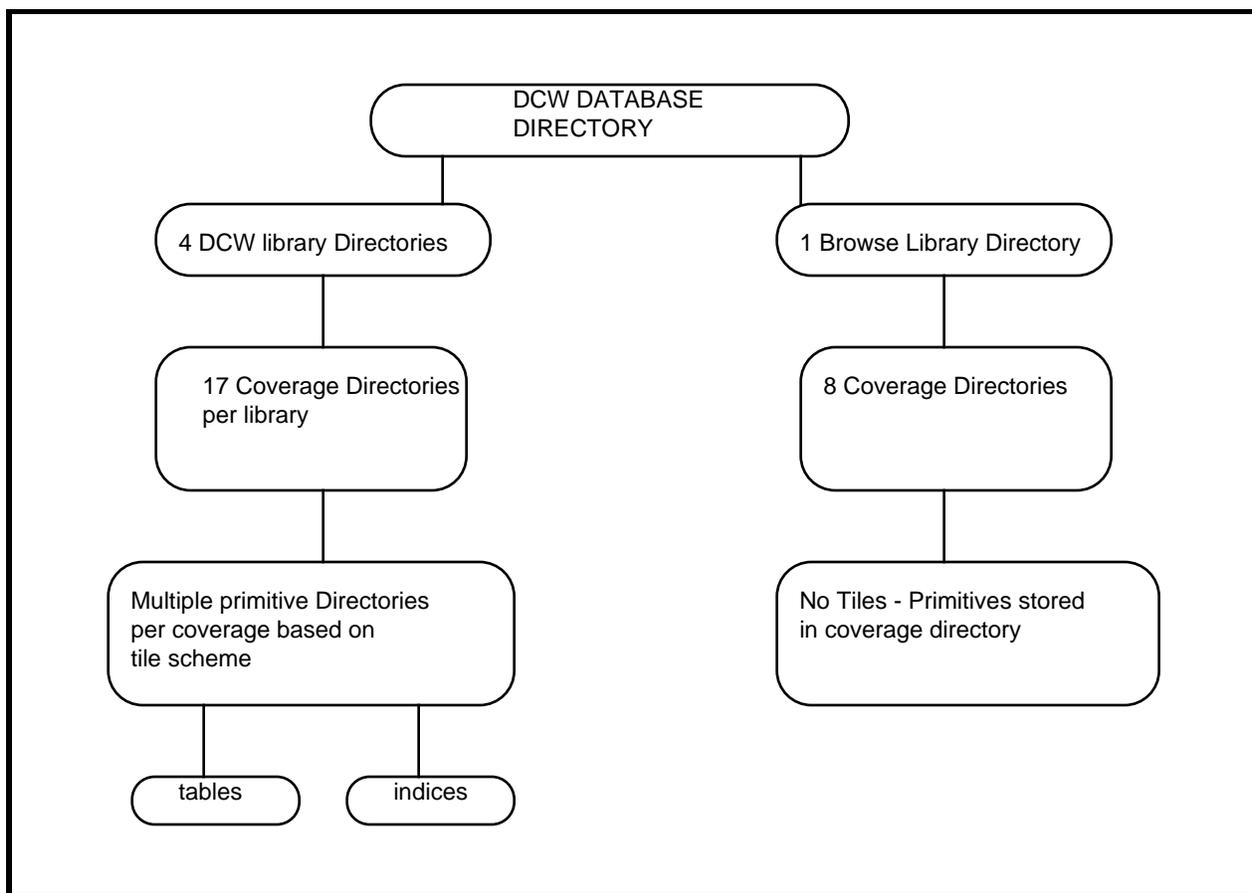


Figure D-1. DCW Database Directory

D.2.2 Outline Functionality

D.2.2.1 Outline

As the tool design is at present, the inputs needed to extract land/sea cover flags are as follows.

- a. The parameter name - at present only PO (Political / Oceans)
- b. The number of parameters - at present only 1
- c. The longitude of the point(s) - in the form +/- 180.0000; e.g. 134.2234
- d. The latitude of the point(s) - in the form +/- 90.0000 e.g. 87.8945
- e. The number of points - 1 or more
- f. A results array already specified by the user. (This will be filled up by the tool)

E.G.

```
PGS_AA_dcw ('po', 1, 34.222, 87.8923, 1, [100][10]);
```

The tool looks at each **long/lat** pair in turn, and searches the database. The first hurdle the tool encounters is the set up of the DCW database. The world has been divided into four areas:

- Europe and northern Asia
- South America, Africa and Antarctica
- North America
- Southern Asia and Australia

To find the relevant location; and extract the data base value; the tool works in the following way.

- a. Locate within which continent the search point lies
- b. Locate the table containing the search point.

NOTE: There may be cases within the Database where the point lies on the junction of two edges; and because of machine accuracy and scale issues, the database will provide no return to the search. If this happens the search is performed again with the addition of a value that will not alter the search due to scalar issues, but will move the point away from the junction so a value can be extracted.

- c. Open the relevant table.
- d. Locate the search point within the table.
- e. Extract the value pertaining to that search point.
- f. Close the table.

- g. Return the result of the search.
- h. Perform another search using the next input coordinate pair.

D.2.3 Optimal Operation

Optimal operation for extraction of data from the data base is accomplished at present by running the tool as stated above. The tool can be run in two modes. The first is calling the tool with one point at a time, the second being calling the tool once with all the points needed as inputs. Of the two the latter is the fastest.

D.2.4 Upgrades

D.2.4.1 Access Speed

At present the tool goes through the above process for every location, provided by the user, as can be expected this will slow down the search process and tool performance. There is a mechanism by which the tool can be speeded up - which may be implemented at a later date, and involves using the file headers in a more constructive fashion. Within the header, there is information about the adjoining tiles. Since most users will be using this tool in a swath based format, the tool will become more time efficient by staying down at the table level, and utilizing code to extract adjoining tile identifiers - rather than performing the search criterion for every single search location.

D.2.4.2 Additional Coverages

The tool has been developed in such a way, that if requirements for other coverages i.e., vegetation, drainage, hypsography are needed - all that is needed is for the data to be supplied, and an additional small code change made to facilitate the new parameters. The results array will then be filled up with integer values representing the vegetation, drainage, etc., type to found at the location provided by the user.

D.3 PGS_AA_dem, PGS_AA_2DRead, PGS_AA_2Dgeo, PGS_AA_3DRead, PGS_AA_3Dgeo

D.3.1 Data Sets Accessed.

D.3.1.1 Introduction

These tools are designed to give access to a wide range of data sets all having all of the following characteristics

- gridded (i.e. raster or cell structured), with parameter value or values associated with each cell constituting the substance of the data set.
- rectangular, having 2 or 3 dimensions

- formatted in simple binary or ASCII with (in C terms) char, float or double, short or long integers aligned to byte boundaries.
- the physical data set is sufficiently small to be loaded into machine memory.

The latter two of these points are involved with pre-processing and implementation issues respectively and are dealt with later (3.3.3.).

Several data sets have been delivered with the toolset. These data sets were considered useful for testing purposes and may also satisfy some science team requirements. They were obtained from NOAA's National Geophysical Data Center in Boulder, Colorado. They are described in outline below. Further details are found in the delivered format and support files (described below). Full details are found in the National Geophysical Data Center (NGDC) publications:

Global Ecosystems Database Version 1.0 (on CD-ROM) User's Guide EPA/600/R-92/194a

Global Ecosystems Database Version 1.0 (on CD-ROM) Documentation Manual (Disc-A) EPA/600/R-92/194b

Global View 4 CD-ROM set. United States Department of Commerce (USDC), National Oceanographic and Atmospheric Administration (NOAA), National Environmental Satellite Data and Information Service (NESDIS), National Geophysical Data Center (NGDC), Boulder Colorado.

Table D-1. Data Included in ToolkiK3/4/5 (1 of 2)

| Data Set | Units | Cell size | File |
|--|--------------|------------------|-----------------|
| Olson World Ecosystems v1.3a | 30 cats | 30 arc min | owe13a.img |
| Olson World Ecosystems v1.4d | 74 cats | 10 arc min | owe14d.img |
| Olson World Ecosystems v1.4dr | 3 cats | 10 arc min | owe14dr.img |
| Olson (Madagascar) Ecosystems v1.3a | 29 cats | 30 arc min | mowe13a.img |
| Federal Naval Operations Center (FNOC) modal elevation | meters | 10 arc min | fnocmod.imgs |
| FNOC maximum elevation | meters | 10 arc min | fnocmax.imgs |
| FNOC minimum elevation | meters | 10 arc min | fnocmin.imgs |
| FNOC modal elevation | meters | 10 arc min | fnocmod.img_dec |
| FNOC maximum elevation | meters | 10 arc min | fnocmax.img_dec |
| FNOC minimum elevation | meters | 10 arc min | fnocmin.img_dec |
| FNOC primary & 2ndary surface types | 10 cats | 10 arc min | fnocpt.img |
| FNOC ocean/land mask | 2 cats | 10 arc min | fnococm.img |
| FNOC number of ridges | count | 10 arc min | fnocrdg.img |
| FNOC direction of ridges | degrees | 10 arc min | fnocazm.img |
| FNOC water & urban cover | percent | 10 arc min | fnocwat.img |
| Zobler Soil types | 108 cats | 60 arc min | srzsoil.img |
| Zobler associated and included soil units | 279 cats | 60 arc min | srzsubs.img |

Table D–1. Data Included in ToolKit3/4/5 (2 of 2)

| Data Set | Units | Cell size | File |
|---|----------|-------------|-----------------------|
| Zobler associated and included soil units | 279 cats | 60 arc min | srzsubs.img_dec |
| Zobler near surface soil texture | 10 cats | 60 arc min | srztex.img |
| Zobler surface slope | 10 cats | 60 arc min | srzslop.img |
| Zobler soil phase | 87 cats | 60 arc mins | srzphas.img |
| Zobler special codes | 12 cats | 60 arc mins | srzcode.img |
| Zobler world areas | 9 cats | 60 arc mins | srzarea.img |
| Etop05 surface elevation | meters | 5 arc mins | etop05.dat |
| Etop05 surface elevation | meters | 5 arc mins | etop05.dat_dec |
| DMA conterminous USA | meters | 30 arc secs | usatile.bin tiles (3) |
| Terrainbase global DEM (etop05 based) | meters | 5 arc mins | tbase.bin |
| Terrainbase global DEM (etop05 based) | meters | 5 arc mins | tbase .bin tiles (3) |
| Geoid | cm | 15 arc mins | geoid.dat |

- Note 1. The _dec files are byte swapped to allow operation on DEC machines. PGS_AA_dem has a byte swapping utility built in which comes into operation on DEC machines.
- Note 2. The table in section 3.2.2. specifies the parameters names recognized by the tools.
- Note 3. The tiled files are subdivided in order to reduce physical file size.

There is no loss of data. Access to the tiles will yield the same result as to the original whole data set.

These data sets are samples only. Other data sets may be delivered with later versions of the tool kit or the user may use his/her own data sets from other sources.

D.3.1.2 Support and format files

Support and format files are required for each data set. There is one format file per data set but there is not necessarily a one-to-one mapping between data sets and support files since the same support file can be used for similar data sets. The association between these files is specified operationally in the indexFile see section 3.2.2.

The support files for the delivered data sets have been created by the tool developers although in the longer term it is anticipated that users will create their own data sets and support data. These files are simple label = value ASCII files containing a set of values required by the tools.

Format files use the Freeform data description language to describe data file formats. A subset of possible descriptions is accepted by the tool. Full details of Freeform, including format specifications can be found in the Freeform Tutorial accessible on the ftp server

ftp.ngdc.noaa.gov under /pub. Freeform is also a component of the software of the tools. An outline of Freeform data description applicable to the ancillary tools is found below.

D.3.1.2.1 Support File

The support file is constructed using a label = value format and read using the PGS_AA_PeV tool described elsewhere. It contains various values which define the format of the output buffer containing parameter values returned to the user. For 2 and 3 dimensional data sets, there are mandatory fields that must exist in the support file. These are described below with an explanation of how each is derived.

| | |
|------------------|--|
| cacheFormat1 | the data type of the output to be produced by the tool (short, long, double or float). On machines (e.g. sgi IRIX64, dec_alpha) 'long' datatype is eight bytes long. In such cases instead of using 'long', 'int64' must be used. |
| cacheFormat2 | the number of decimal places in the output to be produced by the tool (applicable to double and float only) |
| cacheFormatBytes | the number of bytes represented by the data type of the output |
| parmMemoryCache | the size in bytes of the parameter requested once changed to the output type. The volume of the parameter from the whole data set. |
| dataType | the data type of the output to be produced by the tool (short, long, double or float). |
| autoOperation | a composite integer value made up of operations that must be applied to the data during access (see section 3.2.3) |
| fileMemoryCache | the size in bytes of the data set file in its input format (see format file below) |
| maxLat | maximum latitude of data set |
| minLat | minimum latitude of data set |
| maxLong | maximum longitude of data set |
| minLong | minimum longitude of data set |
| xCells | the number of data set cells in the X (fastest changing) dimension |
| yCells | the number of data set cells in the Y (slower changing) dimension |
| zCells | the number of data set cells in the Z (slowest changing) dimension (set 0 for 2d data sets) |
| funcIndex | index for the interpolation routine to be used. Currently only linear interpolation is supported for which the index is 0. |
| swapBytes | 'yes' to indicate byte swapping is required on the result buffer else 'no'. Used only by the PGS_AA_dem tool on dec machines for cases where the the data files have originated on foreign machines. |

- note 1. parmMemoryCache and fileMemoryCache must be => the appropriate size in bytes
- note 2. the dimensions (*Cells*) must be matched with the storage form of the data set in terms of dimension ordering.

For some data sets, additional support information may be required. The tools will currently deal with the National Meteorological Center (NMC) Rapid Update Cycle (RUC) model products that are in a polar stereographic projection. Thus the following must be present in the support file.

lowerLeftLat of the grid origin
 lowerLeftLong of the grid origin (in E coordinates)
 meshLength length in meters of the cell
 gridOrientation in E coordinates

D.3.1.2.2 Freeform data description

Freeform is able to deal with a number of format types. The data sets delivered with the tool kit are all have relatively simple binary formats described in the '.bfm' files.

e.g. fnocMod 1 2 short 0

- the first item is the parameter name as requested by the user
- the second and third values are the start and stop byte positions of the parameter
- the data type. On machines (e.g. sgi IRIX64, dec_alpha) 'long' datatype is eight bytes long. In such cases instead of using 'long', 'int64' must be used.
- the number of values after the decimal point for float/doubles

These files describe the **input** format of the data set; i.e., the format of the data set; c.f. the **output** format described in the support file that is the format of the buffer delivered to the user through the tool.

The parameter is described once and Freeform assumes the same byte pattern throughout the data file, whatever its size. A data set file may contain multiple parameters with different data types. However, Freeform does not allow multiple parameters to be band interleaved; i.e., multiple parameters must have values individually interleaved, e.g. the format file:

```
fnocMod 1 2 short 0
another_parm 3 6 float 1
```

will allow Freeform to ingest a data set having binary data (when viewed)

```
34 45.3 33 46.1 45 712.3 .....etc.
```

The extension .bfm tells Freeform that the file is in binary format. Other extensions are contents are available in Freeform although the ancillary tools will not deal with them at this release.

D.3.2 Functionality and Operation

D.3.2.1 Outline Functionality

The tools are designed to be called by the user using a parameter name; a file i.d.; an operation; a version number; and either geographic coordinates or file structure coordinates.

The tool takes the **parameter** name and matches it to a list in the indexFile. If found, the file i.d.s of the support and format files are ingested from the indexFile. The format and support files are then interrogated by the tool for relevant information. The **file i.d.** and **version** number provide the full identification for the data set file containing the parameter and must be known by the user from the process control environment (see 3.2.4.).

The **operation** is an integer comprising the sum of operations required by the user to be applied to the data during extraction through the tool. Section 3.2.3 specifies the available operations.

The **geographic coordinates** (input to **PGS_AA_2Dgeo**, **PGS_AA_3Dgeo**) are simple latitude/longitude as double values in the range +/- 180.000 (longitude) and +/-90.000 (latitude). The **file structure coordinates** (x,y and z) (input to **PGS_AA_2DRead**, **PGS_AA_3DRead**) are defined in respect to the ordering of the data in the data set file. The calling sequence expects the 'x' dimension to be the fastest changing dimension followed by 'y' and (for 3 D data sets) 'z'. This means that the user must understand the nature of the ordering of dimensions in the data set file and this should also be reflected in the support file.

Example:

The Olson World Ecosystem Data sets supplied with the tool are ordered with lines of latitude first (i.e., the cells in the binary file start at +90.00, -180.00 and proceed to +90.00, +180.00 before starting the next line of latitude). Thus the value of longitude of each cell changes fastest and so longitude is the x dimension and latitude the y dimension.

Both GEO tools assume that longitude is associated with the fastest changing (x) dimension and perform calculations on this basis. This means the support file xCells value represents the longitude range of the data set. If a data set is oriented with latitude changing fastest then xCells must be set to the number of cells in latitude, and the latitude and longitude input arguments to the calling sequence must be used reversed in meaning; i.e., input user latitude into the longitude argument etc.

PGS_AA_dem operates in a very similar way to **PGS_AA_2Dgeo** that it utilizes. The value added in the DEM tool is that it selects parameter values from the same logical data set where the data are physically separated into tiles. The DEM tool makes the selection on the basis of the maxLat, maxLong, minLat, and minLong attributes found in the Support files.

D.3.2.2 Parameters and the indexFile

The AA tools have been delivered with a sample set of data files. These files contain one parameter only per data file, although the tools will operate with files having multiple parameters (with a limit currently set to 4). The indexFile currently contains parameters found in the sample data sets. **The parameter names in the indexFile are those which must be used in the calling sequences.** When the user wishes to add new data sets, the indexFile must be updated with suitable names for the parameter(s) contained in the data sets plus the i.d.s of the support and format files (i.d.s should cross reference with process control table).

The current indexFile appears as follows:

Table D–2. Current Index File

| Parameter 21 (Number Of Records) | Support File I.D. | Format File I.D. |
|-------------------------------------|-------------------|-------------------|
| OlsonMadagascarEcosystems1.3a | 10901 | 10920 |
| OlsonWorldEcosystems1.3a | 10902 | 10921 |
| OlsonWorldEcosystems1.4d | 10903 | 10922 |
| OlsonWorldEcosystems1.4dr | 10903 | 10923 |
| etop05SeaLevelElevM | 10904 | 10924 |
| fnocAzm | 10905 | 10925 |
| fnocOcm | 10905 | 10926 |
| fnocPt | 10905 | 10927 |
| fnocRdg | 10905 | 10928 |
| fnocSt | 10905 | 10929 |
| fnocUrb | 10905 | 10930 |
| fnocWat | 10905 | 10931 |
| fnocMax | 10906 | 10932 |
| fnocMin | 10906 | 10933 |
| fnocMod | 10906 | 10934 |
| srzArea | 10907 | 10935 |
| srzCode | 10907 | 10936 |
| srzPhas | 10907 | 10937 |
| srzSlop | 10907 | 10938 |
| srzSoil | 10907 | 10939 |
| srzText | 10907 | 10940 |
| nmcRucSigPres | 10909 | 10941 |
| nmcRucSigPot | 10909 | 10941 |
| usadmaelevation | 10740 - 10751 | 10700 - 10711 (2) |
| tbaseElevationWorld | 10915 | 10942 |
| tbaseElevation | 10916 - 10919 | 10943- 10946(2) |
| geoid data | 10948 | 10947 |

Note 1: The nmc file contains 2 parameters of many from a model run for a test period. The are included for test purposes only and are not generally applicable.

Note 2: These elevation parameters cover multiple physical files that are accessed automatically by the DEM tool.

D.3.2.3 Use of User Specified and Auto-Operations

To account for the variability of data sets, two types of 'operation' have been enabled within the tools; user and auto-operations. The **user operation**, the last argument in the calling sequence, specifies which additional functions the user wishes to apply to the data. The currently available operations are:

Operation: PGS_AA_NEARESTCELL
Argument value: 1
Applicable to: PGS_AA_2Dgeo, PGS_AA_3Dgeo
Function:

The geographic coordinates are translated to a column and row coordinate pair. The translation provides a floating point number. Obviously the cell coordinate is an integer. This operation allows the user to specify the nearest cell by rounding the floating point numbers up (using the C 'ceil' function).

Operation: PGS_AA_OP_NINTCELL
Argument value: 2
Applicable to: PGS_AA_3Dgeo
Function:

This operation is specific to the polar stereographic auto-operation the output from which is unclear at the boundary. This user operation is used to round geocoordinate values in a very similar way to PGS_AA_NEARESTCELL but with allowance for uncertain boundary calculations.

Operation: PGS_AA_INTERP2BY2
Argument value: 4
Applicable to: PGS_AA_2Dgeo
Function:

This operation conducts interpolation on a 2x2 grid(i.e. nearest 4 points) and returns the interpolated value. The type of interpolation is controlled by funcIndex defined in the support file. Currently only bilinear interpolation is supported with funcIndex = 0. The interpolation routine was taken from Numerical Recipes in C by William H. Press et al., pages 90 and 106.

Operation: PGS_AA_INTERP3BY3

Argument value: 8
Applicable to: PGS_AA_2Dgeo
Function:

This operation conducts interpolation on a 3x3 grid(i.e. nearest 9 points) and returns the interpolated value. The type of interpolation is controlled by funcIndex defined in the support file. Currently only bilinear interpolation is supported with funcIndex = 0. The interpolation routine was taken from Numerical Recipes in C by William H. Press et al., pages 90 and 106.

Other more complex operations can be conceived although none have been implemented at this time.

Auto-operations are those functions that must be applied in order to extract the correct values. The auto-operation is specified in the support file and applied automatically on each run. The currently available auto-operations are:

Operation: PGS_AA_AOP_PLATTECARRE
Support file value: 1
Applicable to: PGS_AA_2Dgeo, PGS_AA_3Dgeo, PGS_AA_dem
Function:

This auto-operation calculates the column row cell coordinates from geographic coordinates assuming a Platte Carre projection

Operation: PGS_AA_AOP_POLARSTEREO
Support file value: 2
Applicable to: PGS_AA_3Dgeo
Function:

This auto-operation calculates the column row cell coordinates from geographic coordinates assuming an NMC RUC model polar stereographic projection.

Operation: PGS_AA_AOP_GREENWICHSTART
Support file value: 4
Applicable to: PGS_AA_2DRead, PGS_AA_3DRead, PGS_AA_2Dgeo,
PGS_AA_3Dgeo, PGS_AA_dem
Function:

This auto-operation recalculates the geographic coordinates assuming a longitude 0 value at Greenwich.

Operation: PGS_AA_AOP_IDLSTART
Support file value: 8
Applicable to: PGS_AA_2DRead, PGS_AA_3DRead, PGS_AA_2Dgeo,
PGS_AA_3Dgeo, PGS_AA_dem
Function:

This auto-operation recalculates the geographic coordinates assuming a longitude 0 value at the Interactive Data Language (IDL).

Auto-operations are generally applied before user operations.

Both types of operation are additive; e.g., an auto-operation of value 9 will result in the functions PGS_AA_AOP_IDLSTART and PGS_AA_AOP_PLATTECARRE being applied to input geo-coordinates in that order.

D.3.2.4 Operational Environment

The file set i.d. and version number must be provided by the user to the ancillary tool. For a static data set, only the i.d. is relevant, the version number should be set to 1. The i.d. is set up in the process control table during Algorithm Integration and Test (AI&T) of the algorithm and should be known to the user.

For dynamically changing data sets, a version number is required which specifies the exact data file out of a number staged for the processing run (e.g., for a set of times). These are obtained from the process control tools PGS_PC_GetNumberOfFiles and PGS_PC_GetAttributes (described elsewhere in this document). The sequence from calling these tools to obtain a version number is:

```
PGS_PC_GetNumberOfFiles    gets number of versions for a particular i.d.  
LOOP    FOR    number of version with same file i.d.  
    PGS_PC_GetAttributes        of each file version  
    test of attributes using user criterion  
ENDLOOP  
PGS_AA_tool call using i.d. and selected version number
```

This series of calls is the basis of the **PGS_AA_dem** tool that selects the correct tiles using geographic coverage attributes. DEMs or other 2 dimensional data sets that are physically too large to be ingested into RAM in one go, can be 'tiled' into smaller coverages. These are then entered into the PCF having the same fileId but different version numbers. The PGS_AA_dem tool makes the selection and fills the results buffer for the user.

D.3.3 Optimal Operation

D.3.3.1 Buffering

The tools ingest the whole data file into a buffer and then extract the parameter required into a further parameter buffer. The area requested is then extracted and returned in the output/results buffer. The parameter buffer is "free'd" before exiting the tool. This leaves the file buffer in memory. Subsequent calls requesting parameter values from the same file are serviced from this buffer while parameters from other files obviously cause the new file to be buffered. There is a user configurable number of file buffers which can be held by each tool. It should be set by the user according to the memory limitation of the host machine and the need for rapid access.

Obviously, the greater the number of files held, the quicker different parameter calls will be serviced, but at the expense of tying up memory. The #define is currently set to 4 in PGS_AA.h (FORTRAN version is PGS_AA.f):

```
#define PGSd_AA_MAXNOCACHES 4
```

D.3.3.2 Multiple calls

The GEO tools can be used with single coordinate pairs repeatedly; e.g., calling the tool in a loop with changing lat/longs. The tools can also accept arrays of coordinate pairs. Using the tools in this way will illicit a much faster response from the tool since the setup functions called during each tool call are used only once.

D.3.3.3 Pre-processing, formats and file sizes

The static data files delivered with release 1 are in the format provided by the vendor. This format is compatible with Freeform since data set and Freeform development were associated at NGDC. Most of the files are of relatively small size and can readily be loaded into memory. Etop05 is somewhat larger (18 Mbytes) and especially when used with the FORTRAN interface, may demand memory that is not available (or only with virtual swapping).

The FORTRAN problem arises from the fact that only integers of type PGSt_integer which is equivalent to an Integer*4 are permitted. Thus PGS_AA_2DRead is forced to allocate, e.g., 36 Mbytes memory to extract the elevation data during a tool call. This is the principal reason behind tiling larger data sets such as the DEMs.

The ability of Freeform to deal with a range of formats means that pre-processing of many data sets should be minimal. However, data sets that are have a complex internal structure may require more extensive pre-processing. In particular, NMC data sets are multi-dimensional. It is not yet clear whether further tools will need to be developed to deal with these.

D.3.4 Setting up new/user data sets

Users can and are expected to use their own data sets. Below is a check list of the actions that need to be taken when introducing new data sets.

- Check that the data file conforms to the constraints outlined in 3.1.1.
- Construct a Freeform format file and a support file (3.1.2). Check that suitable operations are available and set the auto-operation.
- Edit a suitable file i.d. into the process control table for the data set, the format file and the support file. The latter 2 files must be in the support file section while the data set file i.d. must be in the product input section.
- Edit the indexFile to include a suitable parameter name for parameters in the data set (3.2.2). Include the file i.d.s of the format and support files related to the data set file and as inserted into the process control table.

- Place the data set file in the product input directory and the format and support files in the ~/runtime directory (or equivalent)

D.3.5 Upgrades

D.3.5.1 Interaction with HDF files

Where ancillary inputs are other EOS products, then the format from which the requested data must be extracted may be HDF. Further ancillary tools using HDF libraries may be developed to deal with this scenario.

D.3.5.2 Other format types for user files

Data sets that cannot be dealt with by the current tools may be due to having non-raster (e.g., vector) formats which may necessitate new tools; although possibly continuing to use Freeform. HDF libraries and formats may also be a means of accessing these formats.

D.3.5.3 New Operations

New data sets provided by ECS or the user may require new operations (user and/or auto). Where these are clearly defined and common to several processing chains, then the current tools may be upgraded to include new operations.

D.4 PGS_AA_PeVA

D.4.1 Data Sets accessed

PGS_AA_PeVA is an ancillary tool to be used for performing a parameter equals value extraction. There are three types of extraction that the tool can perform: a string, integer and a real from a parameter input.

The tool will only do this extraction from an ASCII file which the user constructs. An example of a file, is as follows:

```
CACHEFORMAT1 = long
CACHEFORMAT2 = 0
CACHEFORMATBYTES = 4
PARMMEMORYCACHE = 1036800
DATATYPE = long
DATARATE = static
ANEXAMPLEARRAY = (9,5,3,7)
```

D.4.2 Outline Functionality

The tool is designed to be called by the user, using a logical input, a parameter input and returning a value. The logical is an integer whose value is supplied through the PC environment, which gives the i.d. of the file to be acted upon by the PGS_AA_PeVA tool. The parameter is a data set dependent character string produced by the user, and the value returned by the tool is the result of the mapping from the character string to its value.

Example of calling sequence to extract a string called MY_STRING from the logical file 10992, and return the resulting string in MY_STRING_VALUE.

```
PGS_AA_PeVA_string ( 10992, "MY_STRING", MY_STRING_VALUE);
```

The PGS_AA_PeVA tool operates in exactly the same way but allows for arrays to be extracted (see Main User Guide section)

D.4.3 Optimal Operation

There are some restrictions on the format of the data file. All parameter names must be in upper cases. Arrays must be formatted as shown in the example.

The PeV tool is based on Freeform, while the PeVA tool is based on ODL and will therefore produce different types of error conditions.

D.4.4 Upgrades

None anticipated.

This page intentionally left blank.

Appendix E. Example of Level 0 Access Tool Usage

This Appendix gives an end-to-end example of how Level 0 access tools might be used in science software.

As an example, we use CERES processing, insofar as it is understood by ECS at this time. (CERES is chosen for this example because only the TRMM platform has reasonable definition of file formats at this time; LIS L0 processing is similar.) The source document for TRMM formats is "Interface Control Document between the Sensor Data Processing Facility (SDPF) and the Tropical Rainfall Measuring Mission (TRMM) Customers," NASA Mission Operations and Data Systems Directorate, Draft, Nov. 1994. We assume that the TRMM mission specific parameters given in section 10 of that document apply to CERES.

A single normal CERES production run consists of 24 hours of data. For Level 0 processing, there is a single main instrument-specific science dataset, namely science telemetry (Application ID 54). There is also a "housekeeping" file, consisting of various APIDs, which is common to all TRMM instruments. All science data for one 24 hour period is contained in a single file; all other data, including calibration, diagnostic and housekeeping data are contained in a second file. In addition each of these datasets has an associated Detached SFDU (Standard Formatted Data Unit) Header file, which consists of TRMM file metadata.

E.1 Preparing Simulated CERES L0 Files

At the SCF, you must first prepare the input Level 0 data files. You may decide to customize your files by using function `PGS_IO_L0_File_Sim` in a C or FORTRAN program that you code yourself; alternatively you may choose to use the supplied interactive executable driver `LOsim`. The latter method is shown here. The sample given is for creating a science APID file. The housekeeping file generation inputs are slightly different

In the example,

data that you type is given like this;

data generated by program **L0sim** is given like this,

comments and explanations are given like this.

The line

-->

means that you typed a carriage return, so using the default value.

unix% is the UNIX system prompt.

E.1.1 Sample Session

unix% *\$PGSRUN/L0sim*

```
*****
* -----O----- *
*  ___/\_/\_  *
*  ___/  \/\  *
*  /      \_  *
*  /                               *
*  ^^^^^^^^^^^^^^^^^^^^^^^^^^ *
*  ^^^^^^^^^^^^^^^^^^^^^^^^^^ *
*  ^^^^^^^^^^^^^^^^^^^^^^^^^^ *
*  =EOS=                               *
*****
```

ECS L0 FILE SIMULATOR

Enter <return> at a prompt to select the default option (indicated by []). Enter '?' at any prompt for additional information. Enter 'q' at any prompt to quit.

enter spacecraft ID (TRMM, EOS_AM, EOS_PM) [TRMM]:

-->

enter start date in CCSDS ASCII (format A or B)

A) YYYY-MM-DDThh:mm:ss

B) YYYY-DDDThh:mm:ss

enter start date:

-->1997-12-01

enter stop date:

-->1997-12-02T00:00:00

You may leave out the entire time, minutes and seconds, or seconds if desired.

enter time interval in seconds [6.600000 sec]:

-->

enter the desired number of files [1]:

-->

TRMM always has only one file per APID (or housekeeping): EOS AM and PM may have more. Note that you must rerun program *L0sim* for each virtual data set you want, i.e., each "science" APID (or housekeeping); this prompt is asking how many files you want for a given virtual data set.

is this Housekeeping data (y/[n]):

-->

Housekeeping files are special in that they may have many APIDs. If you enter *y* here, you are prompted for the number of APIDs, then APID no. and data length for each APID. In this prototype, APIDs are written APID 1, APID 2, ..., APID *n*, APID 1, APID 2, ... until the stop time you requested is reached.

is this Quicklook data (y/[n]):

-->

For TRMM, the only effect of this input is to set a byte in the file header. For EOS AM and PM, there is no Quicklook data.

enter the Application ID [0]:

-->54

The APID is stamped on each packet. It is also written to the TRMM file header.

enter the Application Data Length [0]:

-->7118

This is the actual length of the packet application data in bytes. It does not include the packet header. All packets for a given APID have the same length.

read in Application Data from file [<none>]:

-->

If you type in the name of a file here, the simulator reads data from this file and writes it into the packet as application data. Here bytes 1–7118 of this file would be written to packet #1, bytes 7119–14238 to packet #2, etc.

specify processing options (y/[n]):

-->

This is for simulating some miscellaneous data in the TRMM file header. It is meant to indicate options applied during SDPF processing, before it gets to ECS.

start date: 1997-12-01T00:00:00

stop date: 1997-12-02T00:00:00

time interval: 6.6000 seconds

This will create approximately 94.65 MB of data.

accept ([y]/n)?

-->

Writing packets out to 1 file:

- start time of next file: 1997–12–01T00:00:00.000000Z
- number of packets in next file: 13091
- writing file: TRMM_G001_1997–12–01T00:00:00Z_V01.DATASET_01 ...
- writing files: TRMM_G001_1997–12–01T00:00:00Z_V01.DATASET_01 ...
TRMM_G001_1997–12–01T00:00:00Z_V01.SFDU_01

The SFDU file is only created for TRMM.

unix%

E.2 CERES Level 0 processing code using the SDP Toolkit

In this section is given an abbreviated example of what CERES L0 processing code might look like. It is assumed here that the datasets will be opened and processed one-at-a-time; this may not be the case in the actual CERES processing. No processing of packet, header or footer data returned is done in this example.

E.2.1 Notes:

The examples show one way of retrieving simulated ephemeris and attitude data corresponding to packet times. For the science file (APID 54), the time of each packet is saved, then later used as input to the Toolkit ephemeris/attitude retrieval tool. To do this, a simulated ephemeris file must have been prepared beforehand. See the Toolkit Primer (Section 7) or Users Guide (Section 6.2.6) for details. (In the production system, this file is assumed to have been created in preprocessing from either Flight Dynamics Facility (FDF) files or from S/C ephemeris packets.)

In the interests of brevity, Detached SFDU Header file processing is completely omitted from the examples, as it is not clear what the information would be used for. Reading and accessing these files would involve use of the tools PGS_PC_GetFileAttr and PGS_PC_GetFileByAttr; see the Toolkit Primer (Section 4) for explanations of these.

Also, to keep things short, no error processing is shown.

The example code is given for illustrative purposes only, and is adapted from an unofficial unit test driver. The code given here has not actually been compiled and tested.

Because there is exactly one physical file per APID (or housekeeping) per day in TRMM L0 data, a virtual data set in Toolkit L0 functions corresponds to a single physical TRMM L0 file. For EOS AM and PM, there may be more than one physical file per given APID; in that case, this code would change, in that one must loop around the GetHeader and GetPacket calls until all physical files are read. There is an example of this in the tool descriptions for these two tools in section 6.2.1.1.

The examples assume the following exists in the PRODUCT INPUT FILES section of the Process Control File (PCF) at the SCF:

```

1|TRMM_G0001_1997-12-01T00:00:00Z_V01.dataset_01|||
      TRMM_G0001_1997-12-01T00:00:00Z_V01.sfd_u_01|1
54|TRMM_G0088_1997-12-01T00:00:00Z_V01.dataset_01|||
      TRMM_G0088_1997-12-01T00:00:00Z_V01.sfd_u_01|1

```

(Note: each entry must appear on one line in the actual PCF, and not be broken into two lines as shown here.)

C code example

```

#include <PGS_IO.h>
#include <PGS_TD.h>

/* File logicals corresponding to PCF entries
   Arbitrarily use APID as file logical, or 1 for housekeeping */
#define HOUSEKEEPING 1
#define SCIENCE 54

/* PACKET_BUFFER_MAX is the maximum possible size of a telemetry packet,
   including packet header. Note that the input to L0sim corresponding to
   this is "Application Data Length"; however, the latter does not include
   packet header. Since the packet header is 14 bytes for TRMM, we used the
   value 7118 for the "Application Data Length" field in constructing the
   simulated files above. */
#define PACKET_BUFFER_MAX 7132

/* HEADER_BUFFER_MAX is the maximum possible size of the TRMM file header.
   This number is 26 for EOS AM and PM, since those file headers have no
   variable length part. */
#define HEADER_BUFFER_MAX 556

/* FOOTER_BUFFER_MAX is the maximum possible size of the TRMM file "footer,"
   which consists of Quality and Accounting Capsule (QAC) and optionally
   Missing Data Unit List (MDUL). This number is a wild guess. */
#define FOOTER_BUFFER_MAX 100000

/* NUM_DATASETS is the number of virtual datasets to process.
   This includes the housekeeping file and the science file. */
#define NUM_DATASETS 2

/* MAX_PKTS is the maximum number of packets.
   Used for saving packet times and for ephemeris and attitude retrieval */
#define MAX_PKTS 14000

main( )
{
PGSt_PC_Logical   file_logical[NUM_DATASETS];
                  /* Logical file ID for PCF */

```

```

PGSt_SMF_status   returnStatus;      /* Toolkit function return value */

PGSt_integeri;   /* Virtual data set loop index */

PGSt_IO_L0_VirtualDataSet
                virtual_file;        /* Virtual file handle */
PGSt_double      start_time;         /* Virtual data set start time */
PGSt_double      stop_time;          /* Virtual data set stop time */

char             asciiUTC_A[28];     /* time in UTC CCSDS ASCII A format */

PGSt_IO_L0_Header header_buffer[HEADER_BUFFER_MAX];
                /* Buffer for receiving header data */
PGSt_IO_L0_Header footer_buffer[FOOTER_BUFFER_MAX];
                /* Buffer for receiving footer data */

PGSt_integerj;   /* Index */
PGSt_integeroffset; /* Offset byte of packet time */

PGSt_scTime      file_time[2][8];   /* File time in PB5 format */
PGSt_double      jdUTC[2];          /* Time in UTC-- Julian date format */
PGSt_booleanonLeap; /* Leap second flag */

PGSt_integerpacket_count; /* No. packets in this file */

PGSt_integerqac_size; /* Size of QAC data in bytes */
PGSt_integermdul_size; /* Size of MDUL data in bytes */

PGSt_integerp;   /* Packets read counter */
PGSt_integerpacket_loop_flag;
                /* Flag for controlling packet read loop */

PGSt_IO_L0_Packet packet_buf[PACKET_BUFFER_MAX];
                /* Buffer for receiving packet data */

PGSt_integerappID; /* Application ID of this packet */
PGSt_integerpkt_seq_count; /* Sequence number of this packet */
PGSt_integerpkt_len; /* Length in bytes of this packet */

PGSt_scTime      pkt_time[MAX_PKTS][8];
                /* Packet time stamps */
PGSt_double      UTC_offset[MAX_PKTS];
                /* packet UTC offset in seconds */

char             asciiUTC_A_eph_start[28];
                /* start time of ephemeris data in UTC CCSDS ASCII A format */
PGSt_double      positionECI[MAX_PKTS][3];
                /* ECI position vectors (m) */
PGSt_double      velocityECI[MAX_PKTS][3];
                /* ECI velocity vectors (m/s) */

```

```

PGSt_double      ypr[MAX_PKTS][3]; /* Euler angles (yaw/pitch/roll) (rad) */
PGSt_double      yprRate[MAX_PKTS][3];
                                /* Euler angle rates (rad/sec) */
PGSt_double      attitQuat[MAX_PKTS][4];
                                /* Attitude quaternions */

/*****
/* For each data set (housekeeping or "science" APID)
*****/

file_logical[0] = HOUSEKEEPING;
file_logical[1] = SCIENCE;

for( i=0; i<NUM_DATASETS; i++)
{

/*****
/* Call PGS_IO_L0_Open to get a virtual file handle,
/*      start and stop times of the available data
*****/

    returnStatus = PGS_IO_L0_Open( file_logical[i], TRMM,
        &virtual_file, &start_time, &stop_time);

/*****
/* Translate times to ASCII in case you want to print them out or do
/*      something similar
*****/

    returnStatus = PGS_TD_TAItoUTC(start_time,asciiUTC_A);
    returnStatus = PGS_TD_TAItoUTC(stop_time,asciiUTC_A);

/*****
/* Call PGS_IO_L0_SetStart to position the file pointer at 20 minutes after
/*      data start
*****/

    returnStatus = PGS_IO_L0_SetStart( virtual_file, start_time+1200. );

/*****
/* Call PGS_IO_L0_GetHeader to retrieve header and footer
/*      information from the physical file
*****/

    returnStatus = PGS_IO_L0_GetHeader( virtual_file,
        HEADER_BUFFER_MAX, header_buffer,
        FOOTER_BUFFER_MAX, footer_buffer );

```

```

/*****
/*  Unpack and/or save or process header data here
*****/

/*
Header buffer contents:
Bytes 1- 2 : 6 bits spare, 10 bits S/C ID
Bytes 3-11 : S/C clock start time (PB5 format)
Byte    12 : spare
Bytes 13-21 : S/C clock stop time (PB5 format)
Byte    22 : spare
Bytes 23-26 : No. packets in file
*/

/*
Convert S/C time to ASCII, in case you want to print it
*/

for(j=0;j<8;j++)
{
    file_time[0][j] = header_buffer[ 2+j]; /* start */
    file_time[1][j] = header_buffer[12+j]; /* stop */
}
for(j=0;j<2;j++)
{
    returnStatus = PGS_TD_PB5toUTCjd( file_time[j], jdUTC );
    if( returnStatus == PGSTD_N_LEAP_SEC_IGNORED)
    {
        onLeap = PGS_TRUE;
    }
    else
    {
        onLeap = PGS_FALSE;
    }
    PGS_TD_UTCjdtoUTC( jdUTC, onLeap, asciiUTC_A);
}

/* Special notes for EOS AM and PM:
(1) 9th byte of file header time is not used in EOS AM or PM time
    conversions in this prototype
(2) EOS AM and PM file header time format is unknown; we assume they are
    the same as packet time formats. This means that function
    PGS_TD_SCTime_to_UTC must be used to convert EOS AM and PM times to
    ASCII. */

```

```

/*
    Convert no. packets in file to integer
*/

packet_count =
    header_buffer[25] + 256 * (
        header_buffer[24] + 256 * (
            header_buffer[23] + 256 * (
                header_buffer[22] ) ) ) );

/*****/
/* Convert footer sizes to integer: quality (QAC) and missing (MDUL) data
/* (TRMM only)
/*****/

qac_size =
    footer_buffer[3] + 256 * (
        footer_buffer[2] + 256 * (
            footer_buffer[1] + 256 * (
                footer_buffer[0] ) ) ) );
mdul_size =
    footer_buffer[4+qac_size+3] + 256 * (
        footer_buffer[4+qac_size+2] + 256 * (
            footer_buffer[4+qac_size+1] + 256 * (
                footer_buffer[4+qac_size ] ) ) ) );

/*****/
/* Note: the simulator does *not* simulate the internal structure of the QAC
/* and MDUL data
/*****/

/*****/
/* While still packets to process in this file
/*****/

    p = 0;
    packet_loop_flag = 1;
    while( packet_loop_flag )
    {

/*****/
/* Call PGS_IO_L0_GetPacket to read a single L0 packet
/* If reached end of file, set flag to exit loop
/*****/

        returnStatus = PGS_IO_L0_GetPacket(
            virtual_file, PACKET_BUFFER_MAX, packet_buf );
        if ( ( returnStatus == PGSIO_M_L0_HEADER_CHANGED )

```

```

        || ( returnStatus == PGSIO_W_L0_END_OF_VIRTUAL_DS ) )
    {
        packet_loop_flag = 0;
    }

/*****
/*  Unpack and/or save or process packet data
*****/

/*
Packet buffer contents    -- "unused" means not written by simulator
Bytes  1- 2 : packetID    bits 0-2:  Version Number            -- unused
                                bit 3:  Type                    -- unused
                                bit 4:  Secondary Header Flag -- unused
                                bits 5-15: Application Process ID
Bytes  3- 4 : pktSeqCntl  bits 0-1:  Sequence Flags            -- unused
                                bits 2-15: Packet Sequence Count
Bytes  5 -6 : pktLength   Packet Length
Bytes  7-14 : timeStamp   packet S/C time stamp
*/

    appID = packet_buf[1] + 256 * packet_buf[0];
    pkt_seq_count = packet_buf[3] + 256 * packet_buf[2];
    pkt_len = packet_buf[5] + 256 * packet_buf[4];

/* If currently processing the science file (APID 54),
Store time stamps for later retrieval of spacecraft ephemeris

NOTE: Packet time format is spacecraft platform dependent */

    offset = 6; /* 6 for EOS_AM, 7 for EOS_PM */
    if( i == 1)
    {
        for(j=0;j<8;j++)
        {
            pkt_time[p][j] = packet_buf[offset+j];
        }
    }

    p++;
} /* End while (packet_Loop_flag) */

/*****
/* Call PGS_IO_L0_Close to close the virtual data set
*****/

returnStatus = PGS_IO_L0_Close(virtual_file);

```

```

/*****/
/* If currently processing the science file (APID 54),
/* Retrieve simulated S/C ephemeris and attitude at packet times
/* from previously prepared ephemeris file
/*****/

    if( i == 1)
    {
        returnStatus = PGS_TD_SCTime_to_UTC( TRMM, pkt_time, p, asciiUTC_A,
UTC_offset );

        returnStatus = PGS_EPH_EphemAttit( TRMM, asciiUTC_A, UTC_offset,
            PGS_TRUE, PGS_TRUE, asciiUTC_A_eph_start,
            positionECI, velocityECI, ypr, yprRate, attitQuat );
    }

/*****/
/* End for (each data set)
/*****/
}
}

```

FORTTRAN code example

```

implicit none

INCLUDE      'PGS_SMF.f'
INCLUDE      'PGS_PC.f'
INCLUDE      'PGS_PC_9.f'
INCLUDE      'PGS_TD.f'
INCLUDE      'PGS_IO.f'
INCLUDE      'PGS_IO_1.f'

integer      NUM_DATASETS
parameter    (NUM_DATASETS=2)

integer      pgs_mem_calloc
integer      pgs_io_l0_open
integer      pgs_td_taitoutc
integer      pgs_io_l0_setstart
integer      pgs_io_l0_getheader
integer      pgs_td_pb5toutcjd
integer      pgs_td_utcjdtoutc
integer      pgs_io_l0_getpacket
integer      pgs_io_l0_close
integer      pgs_td_sctime_to_utc
integer      pgs_eph_ephemattit

```

```

integer          file_logical(2)
integer          i

integer          returnstatus
integer          virtual_file
double precision start_time
double precision stop_time

character*27asciiutc_a

character*556    header_buffer
character*100000 footer_buffer

integer          j
character*8      file_time(2)
double precision jdutc(2)
integer          onleap

integer          packet_count

integer          qac_size
integer          mdul_size

integer          packet_loop_flag

character*7132   packet_buf

integer          appid
integer          pkt_seq_count
integer          pkt_len
integer          offset

character*8      pkt_time(14000)
double precision utc_offset(14000)

character*27asciiutc_a_eph_start

double precision eciposition(3,14000)
double precision ecivelocity(3,14000)
double precision ypr(3,14000)
double precision yprrate(3,14000)
double precision attitquat(4,14000)

```

```

C *****/
C For each data set (housekeeping or science APID)
C *****/

file_logical(1) = 1
file_logical(2) = 54

do 10 i=1,NUM_DATASETS

```

```

C *****/
C Call pgs_io_l0_open to get a virtual file handle,
C   start and stop times of the available data
C *****/
    returnstatus = pgs_io_l0_open( file_logical(i), TRMM, virtual_file,
        start_time,
        stop_time)

C *****/
C Translate times to ASCII in case you want to print them out or do something
C   similar
C *****/
    returnstatus = pgs_td_taitoutc(start_time,asciiutc_a)
    returnstatus = pgs_td_taitoutc(stop_time,asciiutc_a)

C *****/
C Call pgs_io_l0_setstart to position the file pointer at 20 minutes after
C   data start
C *****/
    returnstatus = pgs_io_l0_setstart( virtual_file, start_time+1200. )

C *****/
C Call pgs_io_l0_getheader to retrieve header and footer
C   information from the physical file
C *****/
    returnstatus = pgs_io_l0_getheader( virtual_file, 556, header_buffer,
        100000, footer_buffer )

C *****/
C   Unpack and/or save or process header data here
C *****/

C
C   Header buffer contents:
C   Bytes 1- 2 : 6 bits spare, 10 bits S/C ID
C   Bytes 3-11 : S/C clock start time (PB5 format)
C   Byte   12 : spare
C   Bytes 13-21 : S/C clock stop time (PB5 format)
C   Byte   22 : spare
C   Bytes 23-26 : No. packets in file
C

```

C Convert S/C start and stop time to ASCII, in case you want to print it

```
do 20 j=1,8
  file_time[1] = header_buffer(3:11)
  file_time[2] = header_buffer(13:21)
20 continue

do 30 j=1,2
  returnstatus = pgs_td_pb5toutcjd( file_time(j), jdutc )
  if( returnstatus .eq. PGSTD_N_LEAP_SEC_IGNORED) then
    onLeap = PGS_TRUE
  else
    onLeap = PGS_FALSE
  end if
  pgs_td_utcjdtoutc( jdutc, onleap, asciiutc_a)
30 continue
```

C Special notes for EOS AM and PM:

C (1) 9th byte of file header time is not used in EOS AM or PM time
C conversions in this prototype
C (2) EOS AM and PM file header time format is unknown we assume they are
C the same as packet time formats. This means that function
C pgs_td_sctime_to_utc must be used to convert EOS AM and PM times to
C ASCII.

C
C Convert no. packets in file to integer
C

```
packet_count =
.      header_buffer(26) + 256 * (
.      header_buffer(25) + 256 * (
.      header_buffer(24) + 256 * (
.      header_buffer(23) )))
```

C *****/

C Unpack footer sizes: quality (QAC) and missing (MDUL) data (TRMM only)

C *****/

```
qac_size =
.      footer_buffer(4) + 256 * (
.      footer_buffer(3) + 256 * (
.      footer_buffer(2) + 256 * (
.      footer_buffer(1) )))
mdul_size =
.      footer_buffer(4+qac_size+4) + 256 * (
.      footer_buffer(4+qac_size+3) + 256 * (
```

```

.          footer_buffer(4+qac_size+2) + 256 * (
.          footer_buffer(4+qac_size+1) )))
C *****/
C   Note: the simulator does *not* simulate the internal structure of the QAC
C   and MDUL data
C *****/
C *****/
C   While still packets to process in this file
C *****/

    p = 1
    packet_loop_flag = 1

    do while( packet_loop_flag .eq. 1 )
C *****/
C   Call PGS_IO_L0_GetPacket to read a single L0 packet
C   If reached end of file, set flag to exit loop
C *****/

        returnStatus = pgs_io_l0_getpacket( virtual_file, 7132, packet_buf
)

        if ( ( returnStatus .eq. PGSIO_M_L0_HEADER_CHANGED )
.          .or. ( returnStatus .eq. PGSIO_W_L0_END_OF_VIRTUAL_DS ) )
then
            packet_loop_flag = 0
        end if

C *****/
C   Unpack and/or save or process packet data
C *****/

C
C   Packet buffer contents    -- "unused" means not written by simulator
C   Bytes  1- 2 : packetID    bits 0-2:  Version Number          -- unused
C                                     bit 3:      Type              -- unused
C                                     bit 4:      Secondary Header Flag -- unused
C                                     bits 5-15:  Application Process ID
C   Bytes  3- 4 : pktSeqCntl  bits 0-1:  Sequence Flags          -- unused
C                                     bits 2-15:  Packet Sequence Count
C   Bytes  5 -6 : pktLength   Packet Length
C   Bytes  7-14 : timeStamp   packet S/C time stamp
C

```

```

        appID = packet_buf(2) + 256 * packet_buf(1)
        pkt_seq_count = packet_buf(4) + 256 * packet_buf(3)
        pkt_len = packet_buf(6) + 256 * packet_buf(5)

C   If currently processing the science file (APID 54),
C       Store time stamps for later retrieval of spacecraft ephemeris
C       NOTE: Packet time format is spacecraft platform dependent

        if( i .eq. 2 ) then
            offset = 7
            pkt_time(p) = packet_buf(offset:14)
40         offset
        end if

        p = p + 1

C   End while (packet_loop_flag)

        end do

C *****/
C   Call PGS_IO_L0_Close to close the virtual data set
C *****/

        returnstatus = pgs_io_l0_close(virtual_file)

C *****/
C   If currently processing the science file (APID 54),
C       Retrieve simulated S/C ephemeris and attitude at packet times
C       from previously prepared ephemeris file
C *****/

        if( i .eq. 2) then
            returnstatus = pgs_td_sctime_to_utc( TRMM, pkt_time, p,
                asciutc_a, utc_offset )

            returnstatus = pgs_eph_ephemattit( TRMM, asciutc_a, utc_offset,
                PGS_TRUE, PGS_TRUE, asciutc_a_eph_start,
                .
                .
                positioneci, velocityeci, ypr, yprate, attitquat )
        end if

C *****/
C   End for (each data set)
C *****/

10    continue

```

Appendix F. Level 0 File Formats

This Appendix gives the definition of file formats assumed in construction of the Level 0 access tools, **PGS_IO_L0_***, and the file simulator **L0sim**. See section 6.2.1.1.

Notes on table entries:

- "Y" in the SIM? column means that this value is simulated by the L0sim software; no entry means that the value is either 0 or garbage in the simulated file.
- No entry in the BIT column means bits 1_8.

F.1 Tropical Rainfall Measuring Mission (TRMM) File Formats

The source document for the TRMM file format is "Interface Control Document between the Sensor Data Processing Facility (SDPF) and the Tropical Rainfall Measuring Mission (TRMM) Customers," NASA Mission Operations and Data Systems Directorate, Draft, Nov. 1994. We assume that the TRMM mission specific parameters given in section 10 of that document apply to CERES and LIS.

TRMM has 2 files associated with each "science" APID or housekeeping file; a detached SFDU header file, an ASCII text file consisting of file metadata, and the main data file.

F.1.1 TRMM Files Schematic

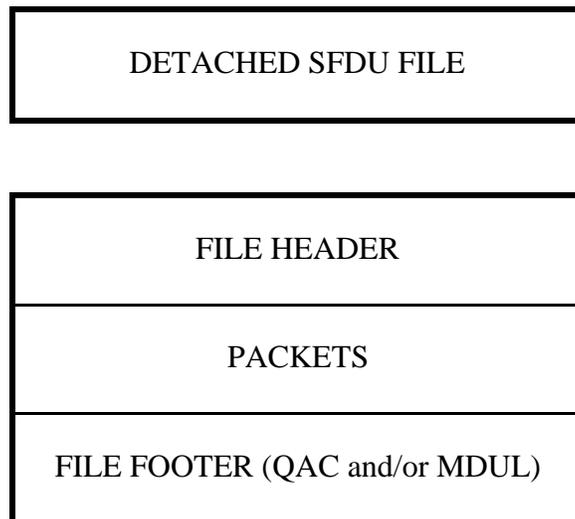


Figure F-1. TRMM Files Schematic

There is one pair of these files for each "science" APID, plus one pair for housekeeping. CERES has 3 "science" APIDs, thus will have 4 pairs of these files per day; LIS has one "science" APID, so will have 2 pairs per day.

F.1.2 Detached SFDU File

This is an ASCII text file containing file metadata. The format of this file is defined in the source document "Interface Control Document between the Sensor Data Processing Facility (SDPF) and the Tropical Rainfall Measuring Mission (TRMM) Customers," NASA Mission Operations and Data Systems Directorate, Draft, Nov. 1994, section 3.2.2.

Note: The Spacecraft Clock time format used in the file header is different from the format used for the packet Time Stamp.

F.1.3 TRMM File Header

Table F-1. TRMM File Header

| BYTE | BIT | PARAMETER | SIM? |
|-------|-----|--|------|
| 1 | 1-6 | (reserved) | |
| | 7-8 | Spacecraft ID | |
| 2 | | Spacecraft ID | Y |
| 3-11 | | Spacecraft Clock - first packet (PB5, microsec accuracy) | Y |
| 12 | | (spare) | |
| 13-21 | | Spacecraft Clock - last packet (PB5, microsec accuracy) | Y |
| 22 | | (spare) | |
| 23-26 | | Number of packets in file | Y |
| 27 | | Processing Options | Y |
| 28 | | Data Type Flag | Y |
| 29-35 | | Time of Receipt at Originating Node (PB5, msec accuracy) | Y |
| 36-38 | | (spare) | |
| 39 | | Select Options | Y |
| 40 | | Number of APIDs | Y |
| 41-42 | | APID | Y |
| 43 | | (spare) | |
| 44 | | Number of QAC lists in File | Y |
| 45-48 | | Offset to QAC list | Y |

Byte numbers are shown for a "science" file.

Byte 2, Spacecraft ID, is always 6b (hex).

Byte 27, Processing Options:

bit 3 on, Redundant Data Deleted
 bit 6 on, Data Merging
 bit 7 in, RS Decoding

Byte 28, Data Type Flag:

=1, Routine Production Data
 =2, Quicklook Data

Note: Routine production and quicklook files have the same format.

Bytes 29–35, Time of Receipt at Originating Node, is arbitrarily set to be equal to

Spacecraft Clock - last packet (without microseconds).

Byte 39, Select Options, is always 2, to indicate data organized by APID

Byte 40, Number of APIDs

=1, "Science" file
 >1, Housekeeping file

Bytes 41–42 are repeated for each APID in a housekeeping file.

Byte 44, Number of QAC lists in File, is always 1.

Bytes 45–48, Offset to QAC list, is measured in bytes from the last byte of this

field to the QAC footer start. Equal to the total number of bytes in the packet data.

F.1.4 TRMM Packet Data

The source document for the TRMM packet data format is "Tropical Rainfall Measuring Mission (TRMM) Telemetry and Command Handbook, Ó TRMM_490_137, February 21, 1994.

Bytes 1–6 are known as the Primary Packet Header; bytes 7–14 are called the Secondary Packet Header.

Table F–2. TRMM Packet Data

| Byte | Bit | Parameter | Sim? |
|---------|-----|-------------------------------|------|
| 1 | 1–3 | Version Number | Y |
| | 4 | Type | Y |
| | 5 | Secondary Header Flag | Y |
| | 6–8 | Application Process ID (APID) | Y |
| 2 | | Application Process ID (APID) | Y |
| 3 | 1–2 | Sequence Flags | |
| | 3–8 | Packet Sequence Count | Y |
| 4 | | Packet Sequence Count | Y |
| 5–6 | | Packet Length in bytes (=p) | Y |
| 7–14 | | Time Stamp | Y |
| 15-p+14 | | Application Data | Y |

Byte 1, bits 1–3, Version Number, is always 000.
 Byte 1, bit 4, Type, is always 0.
 Byte 1, bit 5, Secondary Header Flag, is always 1.
 Bytes 5–6, Packet Length, is defined as "the length of the entire packet, in bytes, less the length of the primary packet header [6 bytes], less one byte." This is equivalent to the length of the secondary packet header (8 for TRMM) + the length of the application data - 1,

F.1.5 TRMM File Footer

Table F–3. TRMM File Footer Table

| Byte | Bit | Parameter | Sim? |
|-----------|-----|---|------|
| 1–4 | | QAC List Length in bytes (=q) | Y |
| 5-q+4 | | QAC entries | |
| q+5-q+8 | | Missing Data Unit List Length in bytes (=m) | Y |
| q+9-q+m+8 | | Missing Data Unit (MDU) entries | |

QAC and MDU entries are neither simulated nor read in this prototype.

There is no Missing Data Unit List (MDUL) in housekeeping files.

F.2 EOS AM File Formats

F.2.1 EOS AM File Schematic

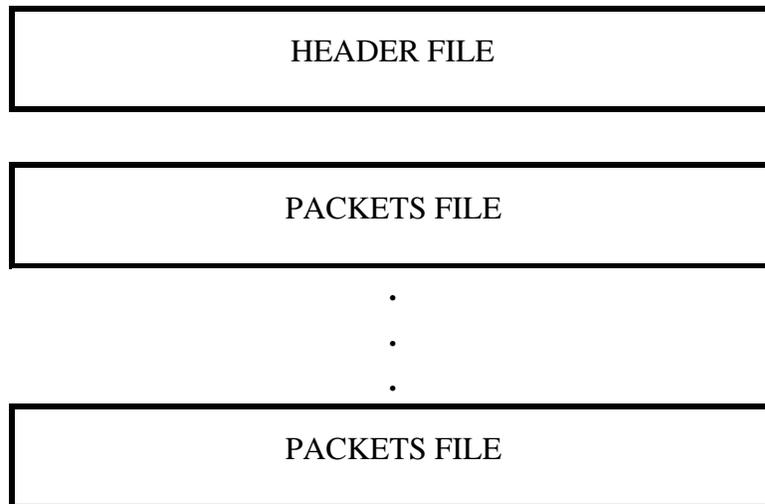


Figure F-2. EOS AM File Schematic

F.2.2 EOS AM File Header

EOS AM L0 data is contained in two or more files: a single header file (Construction Record) and one or more files containing packet data. The actual packet data files have no file header.

For a full description of the EOS AM file header see "Interface Control Document Between The Earth Observing System (EOS) Data and Operations System (EDOS) and the EOS Ground System (EGS) Elements (510-ICD-EDOS/EGS, CDRL B301)", Mission Operations and Data Systems Directorate, Goddard Space Flight Center, January 19, 1996.

F.2.3 EOS AM Packet Data

The source document for the EOS AM packet data format is "Interface Control Document (ICD) Data Format Control Book for EOS-AM Spacecraft (ICD-106)", Martin Marietta IS20008658A, April 19, 1994.

Bytes 1-6 are known as the Primary Packet Header; bytes 7-15 are called the Secondary Packet Header.

Table F-4. EOS AM Packet Data

| Byte | Bit | Parameter | Sim? |
|---------|-----|-------------------------------|------|
| 1 | 1-3 | Version Number | Y |
| | 4 | Type | Y |
| | 5 | Secondary Header Flag | Y |
| | 6-8 | Application Process ID (APID) | Y |
| 2 | | Application Process ID (APID) | Y |
| 3 | 1-2 | Sequence Flags | |
| | 3-8 | Packet Sequence Count | Y |
| 4 | | Packet Sequence Count | Y |
| 5-6 | | Packet Length in bytes (=p) | Y |
| 7 | 1 | Secondary Header ID Flag | Y |
| 7 | 2-8 | Time Stamp | Y |
| 8-14 | | Time Stamp | Y |
| 15 | 1 | Quicklook Flag | |
| 15 | 2-8 | User Flags | |
| 16-p+15 | | Application Data | Y |

Byte 1, bits 1-3, Version Number, is always 000.

Byte 1, bit 4, Type, is always 0.

Byte 1, bit 5, Secondary Header Flag, is always 1.

Bytes 5-6, Packet Length, is defined as "the length of the entire packet, in bytes,

less the length of the primary packet header [6 bytes], less one byte". This is equivalent to the length of the secondary packet header (9 for EOS AM) + the length of the application data - 1,

Byte 7, bit 1, Secondary header ID Flag, is always 0.

Byte 15, bit 1, Quicklook flag: EOS AM quicklook data has been eliminated by NASA.

There is no footer in EOS AM files.

F.3 EOS PM File Formats

F.3.1 EOS PM File Schematic

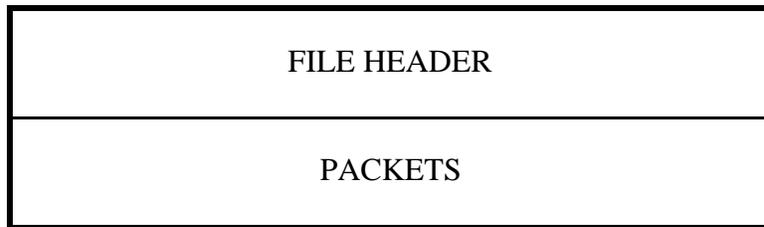


Figure F-3. EOS PM File Schematic

F.3.2 EOS PM File Header

Header format for EOS PM L0 files is unknown at this writing (Feb. 1995).

Arbitrarily we have taken the first 26 bytes of the TRMM file header as the EOS PM file header. Also, since the format of the Spacecraft Clock time in the file header is undefined, we arbitrarily take it as identical to the packet time stamp format.

Table F-5. EOS PM File Header

| Byte | Bit | Parameter | Sim? |
|-------|-----|---------------------------------|------|
| 1 | 1-6 | (reserved) | |
| | 7-8 | Spacecraft ID | |
| 2 | | Spacecraft ID | |
| 3-11 | | Spacecraft Clock - first packet | Y |
| 12 | | (spare) | |
| 13-21 | | Spacecraft Clock - last packet | Y |
| 22 | | (spare) | |
| 23-26 | | Number of packets in file | Y |

F.3.3 EOS PM Packet Data

The source document for the EOS PM packet data format is "General Interface Requirements Document (GIRD) for EOS Common Spacecraft/ Instruments", EOS PM Project, Revision A, GSFC 422_11_12_01, January 1994.

Bytes 1–6 are known as the Primary Packet Header; bytes 7–15 are called the Secondary Packet Header.

Table F–6. EOS PM Packet Data

| Byte | Bit | Parameter | Sim? |
|---------|-----|-------------------------------|------|
| 1 | 1–3 | Version Number | Y |
| | 4 | Type | Y |
| | 5 | Secondary Header Flag | Y |
| | 6–8 | Application Process ID (APID) | Y |
| 2 | | Application Process ID (APID) | Y |
| 3 | 1–2 | Sequence Flags | |
| | 3–8 | Packet Sequence Count | Y |
| 4 | | Packet Sequence Count | Y |
| 5–6 | | Packet Length in bytes (=p) | Y |
| 7 | 1 | Secondary Header ID Flag | |
| 7 | 2 | Quicklook Flag | |
| 7 | 3–8 | User Flags | |
| 8–15 | | Time Stamp | Y |
| 16-p+15 | | Application Data | Y |

Byte 1, bits 1–3, Version Number, is always 000.

Byte 1, bit 4, Type, is always 0.

Byte 1, bit 5, Secondary Header Flag, is always 1.

Bytes 5–6, Packet Length, is defined as "the length of the entire packet, in bytes,

less the length of the primary packet header [6 bytes],
less one byte". This is equivalent to the length of the secondary packet
header (9 for EOS PM) + the length of the application data - 1,

Byte 7, bit 1, Secondary header ID Flag, is always 0.

Byte 15, bit 1, Quicklook flag: EOS PM quicklook data has been eliminated by NASA.

There is no footer in EOS PM files.

F.4 ADEOS-II File Formats

F.4.1 ADEOS-II File Schematic

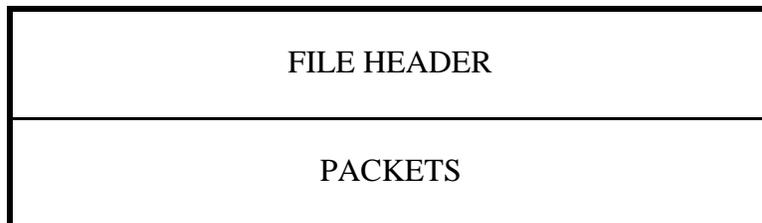


Figure F-4. ADEOS-II File Schematic

F.4.2 ADEOS-II File Header

Header format for ADEOS-II L0 files is unknown at this writing (Feb. 1995).

Arbitrarily we have taken the first 26 bytes of the TRMM file header as the EOS PM file header. Also, since the format of the Spacecraft Clock time in the file header is undefined, we arbitrarily take it as identical to the packet time stamp format.

Table F-7. ADEOS-II File Header

| Byte | Bit | Parameter | Sim? |
|-------|-----|---------------------------------|------|
| 1 | 1-6 | (reserved) | |
| | 7-8 | Spacecraft ID | |
| 2 | | Spacecraft ID | |
| 3-11 | | Spacecraft Clock - first packet | Y |
| 12 | | (spare) | |
| 13-21 | | Spacecraft Clock - last packet | Y |
| 22 | | (spare) | |
| 23-26 | | Number of packets in file | Y |

F.4.3 ADEOS-II Packet Data

The ADEOS-II Packet Data format is preliminary and subject to change (as of 5/15/96).

Bytes 1-6 are known as the Primary Packet Header; bytes 7-15 are called the Secondary Packet Header.

Table F-8. ADEOS-II Packet Data

| Byte | Bit | Parameter | Sim? |
|---------|-----|-------------------------------|------|
| 1 | 1-3 | Version Number | Y |
| | 4 | Type | Y |
| | 5 | Secondary Header Flag | Y |
| | 6-8 | Application Process ID (APID) | Y |
| 2 | | Application Process ID (APID) | Y |
| 3 | 1-2 | Sequence Flags | |
| | 3-8 | Packet Sequence Count | Y |
| 4 | | Packet Sequence Count | Y |
| 5-6 | | Packet Length in bytes (=p) | Y |
| 7-10 | | Instrument Time | Y |
| 11 | | Pulse Time | Y |
| 12-15 | | Orbit Time | Y |
| 16-p+15 | | Application Data | Y |

There is no footer in ADEOS-II files.

This page intentionally left blank.

Appendix G. PGS_GCT Information Relating To Interface Specification

G.1 Projection Id's

PGSd_UTM (Universal Transverse Mercator)
PGSd_ALBERS (Albers Conical Equal Area)
PGSd_LAMCC (Lambert Conformal Conic)
PGSd_MERCAT (Mercator)
PGSd_PS (Polar Stereographic)
PGSd_POLYC (Polyconic)
PGSd_EQUIDC (Equidistant Conic)
PGSd_TM (Transverse Mercator)
PGSd_STEREO (Stereographic)
PGSd_LAMAZ (Lambert Azimuthal Equal Area)
PGSd_AZMEQD (Azimuthal Equidistant)
PGSd_GNOMON (Gnomonic)
PGSd_ORTHO (Orthographic)
PGSd_GVNSP (General Vertical Near-Side Perspective)
PGSd_SNSOID (Sinusoidal)
PGSd_EQRECT (Equirectangular)
PGSd_MILLER (Miller Cylindrical)
PGSd_VGRINT (Van der Grinten)
PGSd_HOM (Hotine Oblique Mercator--HOM)
PGSd_ROBIN (Robinson)
PGSd_SOM (Space Oblique Mercator--SOM)
PGSd_ALASKA (Modified Stereographic Conformal-- Alaska)
PGSd_GOOD (Interrupted Goode Homolosine)
PGSd_MOLL (Mollweide)
PGSd_IMOLL (Interrupted Mollweide)
PGSd_HAMMER (Hammer)
PGSd_WAGIV (Wagner IV)
PGSd_WAGVII (Wagner VII)
PGSd_OBLEQA (Oblated Equal Area)

G.1.1 NOTES

There have been some discrepancies in the output for SOM projection when used for satellites other than LANDSAT. Further investigations led us to the conclusion that the discrepancies were due to a parameter called LANDSAT_RATIO used by the routines. It seemed that the gctpc routines were specifically designed to work for the Landsat satellites.

The documentation of GCTP software says that Landsat Ratio can be an input from the user through projection parameter. But, in fact in the GCTP source code this ratio has been hard coded for Landsat satellite which is 0.5201613.

This ratio causes the grid values to start near the north pole instead of starting at equator at the ascending node. The explanation for this is as follows:

Landsat ratio 0.5201613 comes from the landsat Scene calculations. It seems, in Landsat they divide each orbit into 248 Scenes. They want the starting point to be somewhere at the North Pole and they want it to start at Scene number 64.5 from the ascending node. This number when divided by the number of scenes for half of the globe which is 124 gives you 0.52016129. So by changing this ratio you are changing the start scene for the grid. Setting it to zero makes the grid values to start lets on the equator at the ascending node.

The LANDSAT_RATIO has been renamed as satellite_ratio and the gctpc source code have been modified so that a user can now input the satellite ratio value through the projection parameters. For SOM option B, the satellite ratio is automatically set to 0.5201613.

G.2 GCTP Error Messages

If there is an error in the GCTP freeware library, the tools simply return PGSGCT_E_GCTP_ERROR. However, the actual errors are reported to the LogStatus file using the SMF interface. The list of possible GCTP errors are as follows:

Table G-1. GCTP Error Messages

| Return | Description |
|---------------------------|---|
| PGSGCT_E_STD_PARALLEL | Equal latitudes for St. Parallels on opposite sides of equator |
| PGSGCT_E_ITER_EXCEEDED | Too many iterations in inverse |
| PGSGCT_E_POINT_PROJECT | Point projects into a circle of radius $2 * PI * radius_major$ |
| PGSGCT_E_INPUT_DATA_ERROR | Input data error |
| PGSGCT_E_STD_PARALLEL_OPP | Standard Parallels on opposite sides of equator |
| PGSGCT_E_INFINITY | Point projects into infinity |
| PGSGCT_E_ITER_FAILED | Iteration failed to converge |
| PGSGCT_E_PROJECT_FAILED | Point cannot be projected |
| PGSGCT_E_POINTS_ON_POLES | Transformation cannot be computed at the poles |
| PGSGCT_E_ITER_SOM | 50 iterations without conv |
| PGSGCT_E_SPCS_ZONE | Illegal zone for the given spheroid |
| PGSGCT_E_SPCS_FILE | Error opening State Plane parameter file |
| PGSGCT_E_CONV_ERROR | Convergence Error |
| PGSGCT_E_LAT_15 | Latitude failed to converge after 15 iterations |
| PGSGCT_E_LAT_CONVERGE | Latitude failed to converge |

Table G-2. Projection Transformation Package Projection Parameters (1 of 2)

| Code & Projection Id | Array Element | | | | | | | |
|----------------------|---------------|--------|----------|--------|---------|------------|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 PGSd_UTM | SMajor | SMinor | | | | | | |
| 2 PGSd_SPCS | | | Spheroid | Zone | | | | |
| 3 PGSd_ALBERS | SMajor | SMinor | STDPR1 | STDPR2 | CentMer | OriginLat | FE | FN |
| 4 PGSd_LAMCC | SMajor | SMinor | STDPR1 | STDPR2 | CentMer | OriginLat | FE | FN |
| 5 PGSd_MERCAT | SMajor | SMinor | | | CentMer | LTrueScale | FE | FN |
| 6 PGSd_PS | SMajor | SMinor | | | LongPol | LTrueScale | FE | FN |
| 7 PGSd_POLYC | SMajor | SMinor | | | CentMer | OriginLat | FE | FN |
| 8 PGSd_EQUIDC (A) | SMajor | SMinor | STDPAR | | CentMer | OriginLat | FE | FN |
| PGSd_EQUIDC (B) | SMajor | SMinor | STDPR1 | STDPR2 | CentMer | OriginLat | FE | FN |
| 9 PGSd_TM | SMajor | SMinor | Factor | | CentMer | OriginLat | FE | FN |
| 10 PGSd_STEREO | Sphere | | | | CentLon | CenterLat | FE | FN |
| 11 PGSd_LAMAZ | Sphere | | | | CentLon | CenterLat | FE | FN |
| 12 PGSd_AZMEQD | Sphere | | | | CentLon | CenterLat | FE | FN |
| 13 PGSd_GNOMON | Sphere | | | | CentLon | CenterLat | FE | FN |
| 14 PGSd_ORTHO | Sphere | | | | CentLon | CenterLat | FE | FN |
| 15 PGSd_GVNSP | Sphere | | Height | | CentLon | CenterLat | FE | FN |
| 16 PGSd_SNSOID | Sphere | | | | CentMer | | FE | FN |
| 17 PGSd_EQRECT | Sphere | | | | CentMer | LTrueScale | FE | FN |
| 18 PGSd_MILLER | Sphere | | | | CentMer | | FE | FN |
| 19 PGSd_VGRINT | Sphere | | | | CentMer | OriginLat | FE | FN |
| 20 PGSd_HOM (a) | SMajor | SMinor | Factor | | | OriginLat | FE | FN |
| PGSd_HOM (b) | SMajor | SMinor | Factor | AziAng | AzmthPt | OriginLat | FE | FN |
| 21 PGSd_ROBIN | Sphere | | | | CentMer | | FE | FN |
| 22 PGSd_SOM (a) | SMajor | SMinor | | IncAng | AscLong | | FE | FN |
| PGSd_SOM (b) | SMajor | SMinor | Satnum | Path | | | FE | FN |
| 23 PGSd_ALASKA | SMajor | SMinor | | | | | FE | FN |
| 24 PGSd_GOOD | Sphere | | | | | | | |
| 25 PGSd_MOLL | Sphere | | | | CentMer | | FE | FN |
| 26 PGSd_IMOLL | Sphere | | | | | | | |
| 27 PGSd_HAMMER | Sphere | | | | CentMer | | FE | FN |
| 28 PGSd_WAGIV | Sphere | | | | CentMer | | FE | FN |
| 29 PGSd_WAGVII | Sphere | | | | CentMer | | FE | FN |
| 30 PGSd_OBLEQA | Sphere | | Shapem | Shapen | CentLon | CenterLat | FE | FN |

Table G-2. Projection Transformation Package Projection Parameters (2 of 2)

| Code & Projection Id | Array Element | | | | |
|----------------------|---------------|------|-------|------|------|
| | 9 | 10 | 11 | 12 | 13 |
| 1 PGSd_UTM | | | | | |
| 2 PGSd_SPCS | | | | | |
| 3 PGSd_ALBERS | | | | | |
| 4 PGSd_LAMCC | | | | | |
| 5 PGSd_MERCAT | | | | | |
| 6 PGSd_PS | | | | | |
| 7 PGSd_POLYC | | | | | |
| 8 PGSd_EQUIDC (A) | zero | | | | |
| PGSd_EQUIDC (B) | one | | | | |
| 9 PGSd_TM | | | | | |
| 10 PGSd_STEREO | | | | | |
| 11 PGSd_LAMAZ | | | | | |
| 12 PGSd_AZMEQD | | | | | |
| 13 PGSd_GNOMON | | | | | |
| 14 PGSd_ORTHO | | | | | |
| 15 PGSd_GVNSP | | | | | |
| 16 PGSd_SNSOID | | | | | |
| 17 PGSd_EQRECT | | | | | |
| 18 PGSd_MILLER | | | | | |
| 19 PGSd_VGRINT | | | | | |
| 20 PGSd_HOM (a) | Long1 | Lat1 | Long2 | Lat2 | zero |
| PGSd_HOM (b) | | | | | one |
| 21 PGSd_ROBIN | | | | | |
| 22 PGSd_SOM (a) | PSRev | LRat | PFlag | | zero |
| PGSd_SOM (b) | | | | | one |
| 23 PGSd_ALASKA | | | | | |
| 24 PGSd_GOOD | | | | | |
| 25 PGSd_MOLL | | | | | |
| 26 PGSd_IMOLL | | | | | |
| 27 PGSd_HAMMER | | | | | |
| 28 PGSd_WAGIV | | | | | |
| 29 PGSd_WAGVII | | | | | |
| 30 PGSd_OBLEQA | Angle | | | | |

where

| | |
|---------------|---|
| SMajorSemi | major axis of ellipsoid |
| SMinor | Semi-minor axis of the ellipsoid |
| Spheroid | Used only for state plane projection. Use PGSd_CLARK66 (0) for 1927 datum or GRS80_WGS84(8) for 1983 datum |
| Sphere Radius | of reference sphere. |
| STDPAR | Latitude of the standard parallel |
| STDPR1 | Latitude of the first standard parallel |
| STDPR2 | Latitude of the second standard parallel |
| CentMer | Longitude of the central meridian |
| OriginLat | Latitude of the projection origin |
| FE | False easting in the same units as the semi-major axis |
| FN | False northing in the same units as the semi-major axis |
| LTrueScale | Latitude of true scale |
| LongPol | Longitude down below pole of map |
| Factor | Scale factor at central meridian (Transverse Mercator) or center of projection (Hotine Oblique Mercator) |
| CentLon | Longitude of center of projection |
| CenterLat | Latitude of center of projection |
| Height | Height of perspective point |
| Long1 | Longitude of first point on center line (Hotine Oblique Mercator, format A) |
| Long2 | Longitude of second point on center line (Hotine Oblique Mercator, format A) |
| Lat1 | Latitude of first point on center line (Hotine Oblique Mercator, format A) |
| Lat2 | Latitude of second point on center line (Hotine Oblique Mercator, format A) |
| AziAng | Azimuth angle east of north of center line (Hotine Oblique Mercator, format B) |
| AzmthPt | Longitude of point on central meridian where azimuth occurs (Hotine Oblique Mercator, format B) |
| IncAng | Inclination of orbit at ascending node, counter-clockwise from equator (SOM, format A) |
| AscLong | Longitude of ascending orbit at equator (SOM, format A) |
| PSRev | Period of satellite revolution in minutes (SOM, format A) |
| LRat | Landsat ratio to compensate for confusion at northern end of orbit (SOM, format A -- For LANDSAT, use 0.5201613—See NOTES) |
| PFlag | End of path flag for Landsat: 0 = start of path, 1=end of path (SOM, format A) |
| Satnum | Landsat Satellite Number (1, 2, 3, 4 or 5, SOM format B) |
| Path | Landsat Path Number (Use WRS-1 (World Reference System) for Landsat 1, 2 and 3 and WRS-2 for Landsat4, 5 and 6.) (SOM, format B.) WRS-1 and WRS-2 can be found in Landsat User's Guide. |
| Shapem | Oblated Equal Area oval shape parameter m |
| Shapen | Oblated Equal Area oval shape parameter n |
| angle | Oblated Equal Area oval rotation angle |
| zero | 0 |
| one | 1 |

G.2.1 NOTES

Array elements 14 and 15 are set to zero

All array elements with blank fields are set to zero

All angles (latitudes, longitudes, azimuths, etc.) are in radians

Longitude is negative west of Greenwich

Latitude is negative south of equator

The following notes apply to the Space Oblique Mercator A projection.

A portion of Landsat rows 1 and 2 may also be seen as parts of rows 246 or 247. To place these locations at rows 246 or 247, set the end of path flag (parameter 11) to 1--end of path. This flag defaults to zero.

When Landsat - 1,2,3 orbits are being used, use the following values for specified parameters:

| | |
|--------------|---|
| Parameter 4 | $99^{\circ} 5' 31.2' * \text{PI}/180$ radians |
| Parameter 5 | $128.87 \text{ degrees} - (360/251 * \text{path number}) * \text{PI}/180$ radians |
| Parameter 9 | 103.2669323 |
| Parameter 10 | 0.5201613 |

When Landsat-4,5 orbits are being used, use the following values for the specified parameters:

| | |
|--------------|---|
| Parameter 4 | $99^{\circ} 12' 0' * \text{PI}/180$ radians |
| Parameter 5 | $129.30 \text{ degrees} - (360/233 * \text{path number}) * \text{PI}/180$ radians |
| Parameter 9 | 98.884119 |
| Parameter 10 | 0.5201613 |

*State plane projection is not included in this release. It will be included in the next release.

G.3 UTM Zone Codes

The Universal Transverse Mercator (UTM) Coordinate system uses zone codes instead of specific projection parameters. The table that follows lists UTM zone codes as used by GCTPc Projection Transformation Package. If southern zone is intended then use negative values.

Table G-3. Universal Transverse Mercator (UTM) Zone Codes

| Zone | C.M. | Range | Zone | C.M. | Range |
|------|------|-----------|------|------|-----------|
| 01 | 177W | 180W-174W | 31 | 003E | 000E-006E |
| 02 | 171W | 174W-168W | 32 | 009E | 006E-012E |
| 03 | 165W | 168W-162W | 33 | 015E | 012E-018E |
| 04 | 159W | 162W-156W | 34 | 021E | 018E-024E |
| 05 | 153W | 156W-150W | 35 | 027E | 024E-030E |
| 06 | 147W | 150W-144W | 36 | 033E | 030E-036E |
| 07 | 141W | 144W-138W | 37 | 039E | 036E-042E |
| 08 | 135W | 138W-132W | 38 | 045E | 042E-048E |
| 09 | 129W | 132W-126W | 39 | 051E | 048E-054E |
| 10 | 123W | 126W-120W | 40 | 057E | 054E-060E |
| 11 | 117W | 120W-114W | 41 | 063E | 060E-066E |
| 12 | 111W | 114W-108W | 42 | 069E | 066E-072E |
| 13 | 105W | 108W-102W | 43 | 075E | 072E-078E |
| 14 | 099W | 102W-096W | 44 | 081E | 078E-084E |
| 15 | 093w | 096W-090W | 45 | 087E | 084E-090E |
| 16 | 087W | 090W-084W | 46 | 093E | 090E-096E |
| 17 | 081W | 084W-078W | 47 | 099E | 096E-102E |
| 18 | 075W | 078W-072W | 48 | 105E | 102E-108E |
| 19 | 069W | 072W-066W | 49 | 111E | 108E-114E |
| 20 | 063W | 066W-060W | 50 | 117E | 114E-120E |
| 21 | 057W | 060W-054W | 51 | 123E | 120E-126E |
| 22 | 051W | 054W-048W | 52 | 129E | 126E-132E |
| 23 | 045W | 048W-042W | 53 | 135E | 132E-138E |
| 24 | 039W | 042W-036W | 54 | 141E | 138E-144E |
| 25 | 033W | 036W-030W | 55 | 147E | 144E-150E |
| 26 | 027W | 030W-024W | 56 | 153E | 150E-156E |
| 27 | 021W | 024W-018W | 57 | 159E | 156E-162E |
| 28 | 015W | 018W-012W | 58 | 165E | 162E-168E |
| 29 | 009W | 012W-006W | 59 | 171E | 168E-174E |
| 30 | 003W | 006W-000E | 60 | 177E | 174E-180W |

Obtained from Software Documentation for GCTP general Cartographic Transformation Package: National Mapping Program Technical Instructions, U.S. Geological Survey, National Mapping Division, Oct. 1990,

Note: The following source contains UTM zones plotted on a world map:

Snyder, John P. *Map Projections--A Working Manual*; U.S. Geological Survey Professional Paper 1395

(Supersedes USGS Bulletin 1532), United States Government Printing Office, Washington D.C. 1987. p. 42.

State Plane Coordinate System uses zone codes instead of specific projection parameters. The table that follows lists State Plane Zone Codes as used by the GCTPc Projection Transformation Package.

Table G-4. State Plane Zone Codes (1 of 5)

| Jurisdiction Zone name or number | NAD27 Zone Code | NAD83 Zone Code |
|---|----------------------------|----------------------------|
| Alabama East West | 0101 0102 | 0101 0102 |
| Alaska 01 through 10 thru | 5001 5010 | 5001 5010 |
| Arizona East Central West | 0201 0202 0203 | 0201 0202 0203 |
| Arkansas North South | 0301 0302 | 0301 0302 |
| California 01 through 07 thru | 0401 0407 | 0401 0406 |
| Colorado North Central South | 0501 0502 0503 | 0501 0502 0503 |
| Connecticut | 0600 | 0600 |
| Delaware | 0700 | 0700 |
| District of Columbia | 1900 | 1900 |
| Florida East West North | 0901 0902 0903 | 0901 0902 0903 |

Table G-4. State Plane Zone Codes (2 of 5)

| Jurisdiction Zone name or number | NAD27 Zone Code | NAD83 Zone Code |
|---|--|--|
| Georgia East West | 1001 1002 | 1001 1002 |
| Hawaii 01 through 05 thru | 5101 5105 | 5101 5105 |
| Idaho East Central West | 1101 1102 1103 | 1101 1102 1103 |
| Illinois East West | 1201 1202 | 1201 1202 |
| Indiana East West | 1301 1302 | 1301 1302 |
| Iowa North South | 1401 1402 | 1401 1402 |
| Kansas North South | 1501 1502 | 1501 1502 |
| Kentucky North South | 1601 1602 | 1601 1602 |
| Louisiana North South Offshore | 1701 1702 1703 | 1701 1702 1703 |
| Maine East West | 1801 1802 | 1801 1802 |
| Maryland | 1900 | 1900 |
| Massachusetts Mainland Island | 2001 2002 | 2001 2002 |
| Michigan East (TM) Central (TM) West (TM) North (Lam) Central (Lam) South (Lam) | 2101 2102 2103 2111 2112 2113 | ---- ---- ---- 2111 2112 2113 |

Table G-4. State Plane Zone Codes (3 of 5)

| Jurisdiction Zone name or number | NAD27 Zone Code | NAD83 Zone Code |
|--|----------------------------------|----------------------------------|
| Minnesota North Central South | 2201 2202 2203 | 2201 2202 2203 |
| Mississippi East West | 2301 2302 | 2301 2302 |
| Missouri East Central West | 2401 2402 2403 | 2401 2402 2403 |
| Montana North Central South | ---- 2501 2502 2503 | 2500 ---- ---- ---- |
| Nebraska North South | ---- 2601 2602 | 2600 ---- ---- |
| Nevada East Central West | 2701 2702 2703 | 2701 2702 2703 |
| New Hampshire | 2800 | 2800 |
| New Jersey | 2900 | 2900 |
| New Mexico East Central West | 3001 3002 3003 | 3001 3002 3003 |
| New York East Central West Long Island | 3101 3102 3103 3104 | 3101 3102 3103 3104 |
| North Carolina | 3200 | 3200 |
| North Dakota North South | 3301 3302 | 3301 3302 |
| Ohio North South | 3401 3402 | 3401 3402 |
| Oklahoma North South | 3501 3502 | 3501 3502 |

Table G-4. State Plane Zone Codes (4 of 5)

| Jurisdiction Zone name or number | NAD27 Zone Code | NAD83 Zone Code |
|--|--------------------------------------|--------------------------------------|
| Oregon North South | 3601 3602 | 3601 3602 |
| Pennsylvania North South | 3701 3702 | 3701 3702 |
| Rhode Island | 3800 | 3800 |
| South Carolina North South | ---- 3901 3902 | 3900 ---- ---- |
| South Dakota North South | 4001 4002 | 4001 4002 |
| Tennessee | 4100 | 4100 |
| Texas North North Central Central South Central South | 4201 4202 4203 4204 4205 | 4201 4202 4203 4204 4205 |
| Utah North Central South | 4301 4302 4303 | 4301 4302 4303 |
| Vermont | 4400 | 4400 |
| Virginia North South | 4501 4502 | 4501 4502 |
| Washington North South | 4601 4602 | 4601 4602 |
| West Virginia North South | 4701 4702 | 4701 4702 |
| Wisconsin North Central South | 4801 4802 4803 | 4801 4802 4803 |
| Wyoming East East Central West Central West | 4901 4902 4903 4904 | 4901 4902 4903 4904 |

Table G-4. State Plane Zone Codes (5 of 5)

| Jurisdiction Zone name or number | NAD27 Zone Code | NAD83 Zone Code |
|---|----------------------------------|----------------------------------|
| Puerto Rico | 5201 | 5200 |
| Virgin Islands | ---- | 5200 |
| St. John, St. | 5201 | ---- |
| Thomas | 5202 | ---- |
| St. Croix | | |
| American Samoa | 5300 | ---- |
| Guam | 5400 | ---- |

xxxfor converts input longitude and latitude to the corresponding x,y cartesian coordinates for the xxx projection. The following subroutines follow this general format:

- utmfor (lon, lat, x, y) -- Universal Transverse Mercator (UTM)
- stplnfor (lon, lat, x, y) -- State Plane
- alberfor (lon, lat, x, y) -- Albers
- lamccfor (lon, lat, x, y) -- Lambert Conformal Conic
- merfor (lon, lat, x, y) -- Mercator
- psfor (lon, lat, x, y) --Polar Stereographic
- polyfor (lon, lat, x, y) --Polyconic
- eqconfor (lon, lat, x, y) -- Equidistant Conic
- tmfor (lon, lat, x, y) -- Transverse Mercator (TM)
- sterfor (lon, lat, x, y) -- Stereographic
- lamazfor (lon, lat, x, y) -- Lambert Azimuthal
- azimfor (lon, lat, x, y) -- Azimuthal Equidistant
- gnomfor (lon, lat, x, y) -- Gnomonic
- orthfor (lon, lat, x, y) -- Orthographic
- gvnspffor (lon, lat, x, y) -- General Near Side Perspective
- sinfor (lon, lat, x, y) -- Sinusoidal
- equifor (lon, lat, x, y) -- Equirectangular
- millfor (lon, lat, x, y) -- Miller
- vandgfor (lon, lat, x, y) -- Van Der Grinten
- omerfor (lon, lat, x, y) -- Hotine Oblique Mercator (HOM)
- robfor (lon, lat, x, y) -- Robinson
- somfor (lon, lat, x, y) -- Space Oblique Mercator (SOM)
- alconfor (lon, lat, x, y) -- Alaska Conformal
- goodfor (lon, lat, x, y) -- Goode
- molwfor (lon, lat, x, y) -- Mollweide
- imolwfor (lon, lat, x, y) -- Interrupted Mollweide
- hamfor (lon, lat, x, y) -- Hammer
- wivfor (lon, lat, x, y) -- Wagner IV
- wviifor (lon, lat, x, y) -- Wagner VII
- oblegfor (lon, lat, x, y) -- Oblated Equal Area

xxxinv converts input x,y cartesian coordinates to the corresponding longitude and latitude for the xxx projection. The following subroutines follow this general format:

utminv(x, y, lon, lat) -- Universal Transverse Mercator (UTM)
stplninv(x, y, lon, lat) -- State Plane
alberinv(x, y, lon, lat) -- Albers
lamccinv(x, y, lon, lat) -- Lambert Conformal Conic
merinv(x, y, lon, lat) -- Mercator
psinv(x, y, lon, lat) -- Polar Stereographic
polyinv(x, y, lon, lat) -- Polyconic
eqconinv(x, y, lon, lat) -- Equidistant Conic
tminv(x, y, lon, lat) -- Transverse Mercator (TM)
sterinv(x, y, lon, lat) -- Stereographic
lamazin(x, y, lon, lat) -- Lambert Azimuthal
aziminv(x, y, lon, lat) -- Azimuthal Equidistant
gnominv(x, y, lon, lat) -- Gnomonic
orthinv(x, y, lon, lat) -- Orthographic
gvnspinv(x, y, lon, lat) -- General Near Side Perspective
sininv(x, y, lon, lat) -- Sinusoidal
equiinv(x, y, lon, lat) -- Equirectangular
millinv(x, y, lon, lat) -- Miller
vandginv(x, y, lon, lat) -- Van Der Grinten
omerinv(x, y, lon, lat) -- Hotine Oblique Mercator (HOM)
robinv(x, y, lon, lat) -- Robinson
sominv(x, y, lon, lat) -- Space Oblique Mercator (SOM)
alconinv(x, y, lon, lat) -- Alaska Conformal
goodinv(x, y, lon, lat) -- Goode
molwinv(x, y, lon, lat) -- Mollweide
imolwinv(x, y, lon, lat) -- Interrupted Mollweide
haminv(x, y, lon, lat) -- Hammer
wivinv(x, y, lon, lat) -- Wagner IV
wviiinv(x, y, lon, lat) -- Wagner VII
obleqinv(x, y, lon, lat) -- Oblated Equal Area

This page intentionally left blank.

Appendix H. PGS_CUC_Cons - Example Standard Constants File

Current content of an Example standard constants file

Official file will be supplied by ESDIS Science Office

PI = 3.1415927

ATOMIC_SECOND = 9192631770

MOLECULAR_WEIGHT = 28.970

SOLAR_MOTION_VELOCITY = 19.7

PLANCKS_CONSTANT = 5.6697

This page intentionally left blank.

Appendix I. PGS_CUC_Conv—Input File Provided With the UdUnits Software

This tool uses the UdUnits package to provide unit conversions.

The following information taken from the input file provided with the UdUnits software describes the conversions currently available with the toolkit.

```
# $Id: udunits.dat,v 1.7 1994/02/03 17:20:02 steve Exp $
#
# The first column is the unit name. The second column indicates whether or
# not the unit name has a plural form (i.e., with an 's' appended).
# A 'P' indicates that the unit has a plural form, whereas, a 'S' indicates
# that the unit has a singular form only. The remainder of the line is the
# definition for the unit.
#
# '#' is the to-end-of-line comment-character.
#
# NB: When adding to this table, be *very* careful to distinguish between
# the letter 'O' and the numeral zero '0'. For example, the following two
# entries don't do what one might otherwise expect:
#
#   mercury_0C      mercury_32F
#   millimeter_Hg_0C  mm mercury_0C
#
# BASE UNITS. These must be first and are identified by a nil definition.
#
ampere           P          # electric current
bit              P          # unit of information
candela         P          # luminous intensity
kelvin          P          # thermodynamic temperature
kilogram        P          # mass
meter           P          # length
mole            P          # amount of substance
second          P          # time
radian          P          # plane angle
#
# CONSTANTS
#
percent         S 0.01
```

```

PI                S 3.14159265358979323846
bakersdozen      S 13

%                S percent
pi               S PI

#
# NB: All subsequent definitions must be given in terms of
# earlier definitions. Forward referencing is not permitted.
#
#
# The following are non-base units of the fundamental quantities
#
#
# UNITS OF ELECTRIC CURRENT
#
A                S ampere
amp              P ampere
abampere        P 10 ampere           # exact
gilbert         P 7.957747e-1 ampere
statampere      P 3.335640e-10 ampere
biot            P 10 ampere

#
# UNITS OF LUMINOUS INTENSITY
#
cd              S candela
candle          P candela

#
# UNITS OF THERMODYNAMIC TEMPERATURE
#
degree_Kelvin   P kelvin
degree_Celsius  S kelvin @ 273.15
degree_Rankine  P kelvin/1.8
degree_Fahrenheit P degree_Rankine @ 459.67

#C              S degree_Celsius       # `C' means `coulomb'
Celsius         S degree_Celsius
celsius         S degree_Celsius
centigrade      S degree_Celsius
degC            S degree_Celsius
degreeC         S degree_Celsius
degree_C        S degree_Celsius
degree_c        S degree_Celsius
deg_C           S degree_Celsius

```

| | | |
|---------------------|-------------------------|------------------------|
| deg_c | S degree_Celsius | |
| degK | S kelvin | |
| degreeK | S kelvin | |
| degree_K | S kelvin | |
| degree_k | S kelvin | |
| deg_K | S kelvin | |
| deg_k | S kelvin | |
| K | S kelvin | |
| Kelvin | P kelvin | |
| | | |
| degF | S degree_Fahrenheit | |
| degreeF | S degree_Fahrenheit | |
| degree_F | S degree_Fahrenheit | |
| degree_f | S degree_Fahrenheit | |
| deg_F | S degree_Fahrenheit | |
| deg_f | S degree_Fahrenheit | |
| F | S degree_Fahrenheit | |
| Fahrenheit | P degree_Fahrenheit | |
| fahrenheit | P degree_Fahrenheit | |
| degR | S degree_Rankine | |
| degreeR | S degree_Rankine | |
| degree_R | S degree_Rankine | |
| degree_r | S degree_Rankine | |
| deg_R | S degree_Rankine | |
| deg_r | S degree_Rankine | |
| #R | S degree_Rankine | # `R' means `roentgen' |
| Rankine | P degree_Rankine | |
| rankine | P degree_Rankine | |
| | | |
| # | | |
| # UNITS OF MASS | | |
| # | | |
| assay_ton | P 2.916667e2 kilogram | |
| avoirdupois_ounce | P 2.834952e-2 kilogram | |
| avoirdupois_pound | P 4.5359237e-1 kilogram | # exact |
| carat | P 2e-4 kilogram | |
| grain | P 6.479891e-5 kilogram | # exact |
| gram | P 1e-3 kilogram | # exact |
| kg | S kilogram | |
| long_hundredweight | P 5.080235e1 kilogram | |
| metric_ton | P 1e3 kilogram | # exact |
| pennyweight | P 1.555174e-3 kilogram | |
| short_hundredweight | P 4.535924e1 kilogram | |
| slug | P 14.59390 kilogram | |
| troy_ounce | P 3.110348e-2 kilogram | |

| | | |
|-------------------|------------------------|---------------------------------|
| troy_pound | P 3.732417e-1 kilogram | |
| atomic_mass_unit | P 1.66044e-27 kilogram | |
| tonne | P metric_ton | |
| apothecary_ounce | P troy_ounce | |
| apothecary_pound | P avoirdupois_pound | |
| pound | P avoirdupois_pound | |
| metricton | P metric_ton | |
| gr | S grain | |
| scruple | P 20 grain | |
| apdram | P 60 grain | |
| apounce | P 480 grain | |
| appound | P 5760 grain | |
| atomicmassunit | P atomic_mass_unit | |
| amu | P atomic_mass_unit | |
| t | S tonne | |
| lb | P pound | |
| bag | P 94 pound | |
| short_ton | P 2000 pound | |
| long_ton | P 2240 pound | |
| ton | P short_ton | |
| shortton | P short_ton | |
| longton | P long_ton | |
| # | | |
| # UNITS OF LENGTH | | |
| # | | |
| angstrom | P decinometer | |
| astronomical_unit | P 1.495979e11 meter | |
| fathom | P 1.828804 meter | |
| fermi | P 1e-15 meter | # exact |
| m | S meter | |
| metre | P meter | |
| light_year | P 9.46055e15 meter | |
| micron | P 1e-6 meter | # exact |
| mil | P 2.54e-5 meter | # exact |
| nautical_mile | P 1.852000e3 meter | # exact |
| parsec | P 3.085678e16 meter | |
| printers_pica | P 4.217518e-3 meter | |
| printers_point | P 3.514598e-4 meter | # exact |
| US_statute_mile | P 1.609347e3 meter | # = intn'l mile + .000003 meter |
| US_survey_foot | P 3.048006e-1 meter | |
| chain | P 2.011684e1 meter | |
| inch | S 2.54 cm | # exact |

| | | |
|--------------------|----------------------|---------|
| astronomicalunit | P astronomical_unit | |
| au | S astronomical_unit | |
| nmile | P nautical_mile | |
| nmi | S nautical_mile | |
| inches | S inch | |
| foot | S 12 inch | # exact |
| in | S inch | |
| barleycorn | P inch/3 | |
| ft | S foot | |
| feet | S foot | |
| yard | P 3 foot | |
| furlong | P 660 foot | |
| international_mile | P 5280 foot | # exact |
| arpentlin | P 191.835 foot | |
| yd | S yard | |
| rod | P 5.5 yard | |
| mile | P international_mile | |
| arpentcan | P 27.52 mile | |

#

UNITS OF AMOUNT OF SUBSTANCE

#

| | | |
|-----|--------|--|
| mol | S mole | |
|-----|--------|--|

#

UNITS OF TIME

#

| | | |
|-----------------|---------------------|---------|
| day | P 8.64e4 second | # exact |
| hour | P 3.6e3 second | # exact |
| minute | P 60 second | # exact |
| s | S second | |
| sec | P second | |
| shake | P 1e-8 second | # exact |
| sidereal_day | P 8.616409e4 second | |
| sidereal_minute | P 5.983617e1 second | |
| sidereal_second | P 0.9972696 second | |
| sidereal_year | P 3.155815e7 second | |
| tropical_year | P 3.155693e7 second | |
| year | P 3.153600e7 second | # exact |
| eon | P 1e9 year | |
| d | S day | |
| min | P minute | |
| hr | P hour | |
| h | S hour | |
| fortnight | P 14 day | |

```

yr          P year
a           S year          # "anno"

#
# UNITS OF PLANE ANGLE
#
#rad        P radian        # `rad' means `grey'
circle     P 2 pi radian
angular_degree P (pi/180) radian
turn       P circle
degree     P angular_degree
degree_north S angular_degree
degree_east S angular_degree
degree_true S angular_degree
arcdeg     P angular_degree
angular_minute P angular_degree/60
angular_second P angular_minute/60
grade      P 0.9 angular_degree # exact
degrees_north S degree_north
degreeN    S degree_north
degree_N   S degree_north
degreesN   S degree_north
degrees_N  S degree_north
degrees_east S degree_east
degreeE    S degree_east
degree_E   S degree_east
degreesE   S degree_east
degrees_E  S degree_east
degree_west S -1 degree_east
degrees_west S degree_west
degreeW    S degree_west
degree_W   S degree_west
degreesW   S degree_west
degrees_W  S degree_west
degrees_true S degree_true
degreeT    S degree_true
degree_T   S degree_true
degreesT   S degree_true
degrees_T  S degree_true
arcminute  P angular_minute
arcsecond  P angular_second
arcmin     P arcminute
arcsec     P arcsecond

```

```

#
# The following are derived units with special names. They are useful for
# defining other derived units.
#
steradian      P radian2
hertz          S 1/second
newton         P kilogram.meter/second2
coulomb        P ampere.second
lumen          P candela steradian
becquerel      P 1/second          # SI unit of activity of a
#                                                    # radionuclide
standard_free_fall S 9.806650 meter/second2    # exact

pascal         P newton/meter2
joule          P newton.meter
hz             S hertz
sr             S steradian
force          S standard_free_fall
gravity        S standard_free_fall
free_fall      S standard_free_fall
lux            S lumen/meter2
sphere         P 4 pi steradian
luxes          S lux
watt           P joule/second
gray           P joule/kilogram      # absorbed dose. derived unit
sievert        P joule/kilogram      # dose equivalent. derived unit
mercury_32F    S gravity 13595.065 kg/m3
mercury_60F    S gravity 13556.806 kg/m3
water_39F      S gravity 999.97226 kg/m3  # actually 39.2 F
water_60F      S gravity 999.00072 kg/m3
g              S gravity
volt           P watt/ampere
mercury_0C     S mercury_32F
mercury        S mercury_32F
water          S water_39F
farad          P coulomb/volt
ohm            P volt/ampere
siemens        S ampere/volt
weber          P volt.second
Hg             S mercury
hg             S mercury
H2O            S water
h2o            S water
tesla          P weber/meter2
henry          P weber/ampere

```

```

#
# The following are compound units: units whose definitions consist
# of two or more base units. They may now be defined in terms of the
# preceding units.
#
#
# ACCELERATION
#
gal                P 1e-2 meter/second2 # exact
#
# Area
#
are                P 1e2 m2                # exact
barn               P 1e-28 m2                # exact
circular_mil      P 5.067075e-10 m2
darcy             P 9.869233e-13 m2                # permeability of porous solids
hectare           P 1e4 m2                # exact
acre              P 4840 yard2
#
# ELECTRICITY AND MAGNETISM
#
abfarad           P 1e9 farad                # exact
abhenry           P 1e-9 henry              # exact
abmho             P 1e9 siemens            # exact
abohm             P 1e-9 ohm               # exact
abvolt           P 1e-8 volt              # exact
C                 S coulomb
e                 S 1.6021917e-19 coulomb # charge of electron
chemical_faraday  P 9.64957e4 coulomb
physical_faraday  P 9.65219e4 coulomb
C12_faraday       P 9.64870e4 coulomb
gamma             P 1e-9 tesla             # exact
gauss             S 1e-4 tesla             # exact
H                 S henry
maxwell           P 1e-8 weber             # exact
oersted           P 7.957747e1 ampere/meter
S                 S siemens
statcoulomb       P 3.335640e-10 coulomb
statfarad         P 1.112650e-12 farad
stathenry        P 8.987554e11 henry
statmho           P 1.112650e-12 siemens
statohm           P 8.987554e11 ohm
statvolt          P 2.997925e2 volt

```

| | | |
|-----------|---------------------|-----------------------|
| T | S tesla | |
| unit_pole | P 1.256637e-7 weber | |
| V | S volt | |
| Wb | S weber | |
| mho | P siemens | |
| Oe | S oersted | |
| faraday | P C12_faraday | # charge of 1 mole of |
| # | | # electrons |

#

ENERGY (INCLUDES WORK)

#

| | | |
|------------------------|---------------------|---------|
| electronvolt | P 1.60219e-19 joule | |
| erg | P 1e-7 joule | # exact |
| IT_Btu | P 1.055056 joule | # exact |
| EC_therm | P 1.05506e8 joule | |
| thermochemical_calorie | P 4.184000 joule | # exact |
| IT_calorie | P 4.1868 joule | # exact |
| J | S joule | |
| ton_TNT | S 4.184e9 joule | |
| US_therm | P 1.054804e8 joule | # exact |
| watthour | P watt hour | |
| therm | P US_therm | |
| Wh | S watthour | |
| Btu | P IT_Btu | |
| calorie | P IT_calorie | |
| electron_volt | P electronvolt | |
| thm | S therm | |
| cal | S calorie | |
| eV | S electronvolt | |
| bev | S gigaelectron_volt | |

#

FORCE

#

| | | |
|----------------|--------------------------|---------|
| dyne | P 1e-5 newton | # exact |
| pond | P 1.806650e-3 newton | # exact |
| force_kilogram | S 9.806650 newton | # exact |
| force_ounce | S 2.780139e-1 newton | |
| force_pound | S 4.4482216152605 newton | # exact |
| poundal | P 1.382550e-1 newton | |
| N | S newton | |
| gf | S gram force | |
| force_gram | P 1e-3 force_kilogram | |
| force_ton | P 2000 force_pound | # exact |

| | | | |
|--------------------------------------|---|----------------------------|---------|
| lbf | S | force_pound | |
| ounce_force | S | force_ounce | |
| kilogram_force | S | force_kilogram | |
| pound_force | S | force_pound | |
| ozf | S | force_ounce | |
| kgf | S | force_kilogram | |
| kip | P | 1000 lbf | |
| ton_force | S | force_ton | |
| gram_force | S | force_gram | |
| # | | | |
| # HEAT | | | |
| # | | | |
| clo | P | 1.55e-1 kelvin.meter2/watt | |
| # | | | |
| # LIGHT | | | |
| # | | | |
| lm | S | lumen | |
| lx | S | lux | |
| footcandle | P | 1.076391e-1 lux | |
| footlambert | P | 3.426259 candela/meter2 | |
| lambert | P | (1e4/PI) candela/meter2 | # exact |
| stilb | P | 1e4 candela/meter2 | # exact |
| phot | P | 1e4 lumen/meter2 | # exact |
| nit | P | 1 candela/meter2 | # exact |
| langley | P | 4.184000e4 joule/meter2 | # exact |
| blondel | P | candela/(pi meter2) | |
| apostilb | P | blondel | |
| nt | S | nit | |
| ph | S | phot | |
| sb | S | stilb | |
| # | | | |
| # MASS PER UNIT LENGTH | | | |
| # | | | |
| denier | P | 1.111111e-7 kilogram/meter | |
| tex | P | 1e-6 kilogram/meter | # exact |
| # | | | |
| # MASS PER UNIT TIME (INCLUDES FLOW) | | | |
| # | | | |
| perm_0C | S | 5.72135e-11 kg/(Pa.s.m2) | |
| perm_23C | S | 5.74525e-11 kg/(Pa.s.m2) | |

```

#
# POWER
#
voltampere      P volt ampere
VA              S volt ampere
boiler_horsepower P 9.80950e3 watt
shaft_horsepower P 7.456999e2 watt
metric_horsepower P 7.35499 watt
electric_horsepower P 7.460000e2 watt      # exact
W              S watt
water_horsepower P 7.46043e2 watt
UK_horsepower  P 7.4570e2 watt
refrigeration_ton P 12000 Btu/hour

horsepower      P shaft_horsepower
ton_of_refrigeration P refrigeration_ton

hp              S horsepower

#
# PRESSURE OR STRESS
#
bar              P 1e5 pascal      # exact
standard_atmosphere P 1.01325e5 pascal # exact
technical_atmosphere P 1 kg gravity/cm2 # exact
inch_H2O_39F    S inch water_39F
inch_H2O_60F    S inch water_60F
inch_Hg_32F     S inch mercury_32F
inch_Hg_60F     S inch mercury_60F
millimeter_Hg_0C S mm mercury_0C
footH2O         S foot water
cmHg            S cm Hg
cmH2O           S cm water
Pa              S pascal
inch_Hg         S inch Hg
inch_hg         S inch Hg
inHg            S inch Hg
in_Hg           S inch Hg
in_hg           S inch Hg
millimeter_Hg  S mm Hg
mmHg            S mm Hg
mm_Hg           S mm Hg
mm_hg           S mm Hg
torr            P mm Hg
foot_H2O        S foot water
ftH2O           S foot water

```

| | | |
|--------------------|-----------------------------------|--------------------------|
| psi | S 1 pound gravity/in ² | |
| ksi | S kip/in ² | |
| barie | P 0.1 newton/meter ² | |
| at | S technical_atmosphere | |
| atmosphere | P standard_atmosphere | |
| atm | P standard_atmosphere | |
| barye | P barie | |
| # | | |
| # | # RADIATION UNITS | |
| # | | |
| Bq | S becquerel | |
| curie | P 3.7e10 becquerel | # exact |
| rem | P 1e-2 sievert | # dose equivalent. exact |
| rad | P 1e-2 gray | # absorbed dose. exact |
| roentgen | P 2.58e-4 coulomb/kg | # exact |
| Sv | S sievert | |
| Gy | S gray | |
| Ci | S curie | |
| R | S roentgen | |
| rd | S rad | |
| # | | |
| # | # VELOCITY (INCLUDES SPEED) | |
| # | | |
| c | S 2.997925e+8 meter/sec | |
| knot | P nautical_mile/hour | |
| knot_international | S knot | |
| international_knot | S knot | |
| kt | P knot | |
| # | | |
| # | # VISCOSITY | |
| # | | |
| poise | S 1e-1 pascal second | # absolute viscosity. |
| # | | # exact |
| stokes | S 1e-4 meter ² /second | # exact |
| rhe | S 10/(pascal second) | # exact |
| St | S stokes | |
| # | | |
| # | # VOLUME (INCLUDES CAPACITY) | |
| # | | |
| acre_foot | S 1.233489e3 m ³ | |
| board_foot | S 2.359737e-3 m ³ | |
| bushel | P 3.523907e-2 m ³ | |

| | | |
|------------------------|------------------------|---------------------------------|
| UK_liquid_gallon | P 4.546092e-3 m3 | |
| Canadian_liquid_gallon | P 4.546090e-3 m3 | |
| US_dry_gallon | P 4.404884e-3 m3 | |
| US_liquid_gallon | P 3.785412e-3 m3 | |
| cc | S cm3 | |
| liter | P 1e-3 m3 | # exact. However, from 1901 to |
| # | | # 1964, 1 liter = 1.000028 dm3 |
| stere | P 1 m3 | # exact |
| register_ton | P 3.831685 m3 | |
| US_dry_quart | P US_dry_gallon/4 | |
| US_dry_pint | P US_dry_gallon/8 | |
| US_liquid_quart | P US_liquid_gallon/4 | |
| US_liquid_pint | P US_liquid_gallon/8 | |
| US_liquid_cup | P US_liquid_gallon/16 | |
| US_liquid_gill | P US_liquid_gallon/32 | |
| US_fluid_ounce | P US_liquid_gallon/128 | |
| US_liquid_ounce | P US_fluid_ounce | |
| UK_liquid_quart | P UK_liquid_gallon/4 | |
| UK_liquid_pint | P UK_liquid_gallon/8 | |
| UK_liquid_cup | P UK_liquid_gallon/16 | |
| UK_liquid_gill | P UK_liquid_gallon/32 | |
| UK_fluid_ounce | P UK_liquid_gallon/160 | |
| UK_liquid_ounce | P UK_fluid_ounce | |
| liquid_gallon | P US_liquid_gallon | |
| fluid_ounce | P US_fluid_ounce | |
| #liquid_gallon | P UK_liquid_gallon | |
| #fluid_ounce | P UK_fluid_ounce | |
| dry_quart | P US_dry_quart | |
| dry_pint | P US_dry_pint | |
| liquid_quart | P liquid_gallon/4 | |
| liquid_pint | P liquid_gallon/8 | |
| gallon | P liquid_gallon | |
| barrel | P 42 US_liquid_gallon | # petroleum industry definition |
| quart | P liquid_quart | |
| pint | P liquid_pint | |
| cup | P liquid_gallon/16 | |
| gill | P liquid_gallon/32 | |
| tablespoon | P US_fluid_ounce/2 | |
| teaspoon | P tablespoon/3 | |
| peck | P bushel/4 | |
| oz | P fluid_ounce | |
| floz | S fluid_ounce | |
| acre_feet | S acre_foot | |
| board_feet | S board_foot | |
| Tbl | P tablespoon | |

| | |
|-------|--------------|
| Tbsp | S tablespoon |
| tbsp | S tablespoon |
| Tblsp | S tablespoon |
| tblsp | S tablespoon |
| litre | P liter |
| l | S liter |
| tsp | S teaspoon |
| pk | S peck |
| bu | S bushel |
| fldr | S floz/8 |
| dram | P floz/16 |
| bbl | S barrel |
| pt | S pint |
| dr | S dram |

#

COMPUTERS AND COMMUNICATION

#

| | | |
|------|--------------|---------|
| baud | S 1/second | # exact |
| b | S bit | |
| bps | S bit/second | |
| cps | S hertz | |
| Bd | S baud | |

#

MISC

#

| | | |
|--------------|----------------|---------|
| kayser | P 1e2/meter | # exact |
| rps | S hertz | |
| rpm | S hertz/60 | |
| geopotential | S gravity | |
| work_year | P 2056 hours | |
| work_month | P work_year/12 | |
| gp | S geopotential | |
| dynamic | S geopotential | |

Appendix J. Population of Granule Level Metadata Using the SDP metadata tools

| | |
|---|----|
| 6.2 SDP Toolkit Tools—Mandatory..... | 1 |
| 6.2.1 File I/O Tools..... | 1 |
| 6.2.1.1 Level 0 Science Data Access Tools..... | 1 |
| 6.2.1.2 HDF File Access Tools..... | 2 |
| 6.2.1.4 Metadata..... | 2 |
| 6.2.2 Error/Status Reporting Tools..... | 3 |
| 6.2.3 Process Control Tools..... | 4 |
| 6.2.4 Memory Management Tools..... | 5 |
| 6.2.5 Bit Manipulation Tools..... | 6 |
| 6.2.6 Spacecraft Ephemeris and Attitude Data Access Tools..... | 6 |
| 6.2.7 Time and Date Conversion Tools..... | 6 |
| 6.3 SDP Toolkit Tools - Optional..... | 7 |
| 6.3.1 Ancillary Data Access and Manipulation Tools..... | 7 |
| 6.3.2 Celestial Body Position..... | 8 |
| 6.3.2.1 Celestial Body Access Tools..... | 8 |
| 6.3.3 Coordinate System Conversion..... | 9 |
| 6.3.3.1 Coordinate System Conversion - Transformation Tools..... | 9 |
| 6.3.3.2 Coordinate System Conversion - Other Tools..... | 9 |
| 6.3.4 Geo–Coordinate Transformation Tools..... | 10 |
| 6.3.6 Constants and Unit Conversions..... | 10 |
| B.1 Note..... | 1 |
| B.2 Description..... | 1 |
| C.1 Defining Process Control Files..... | 1 |
| C.1.1 PCF Components..... | 1 |
| C.1.2 Format Rules..... | 2 |
| C.1.3 Format Example..... | 3 |
| C.1.4 Master Template:..... | 5 |

| | |
|--|----|
| C.2 Validating Process Control Files..... | 16 |
| C.2.1 DESCRIPTION:..... | 16 |
| C.2.2 INPUT..... | 17 |
| C.2.3 OUTPUT..... | 17 |
| C.2.4 ERRORS:..... | 18 |
| C.2.5 WARNINGS:..... | 20 |
| C.2.6 EXAMPLES:..... | 20 |
| C.2.6.1 EXAMPLE 1..... | 20 |
| C.2.6.2 EXAMPLE 2..... | 27 |
| C.2.6.3 EXAMPLE 3..... | 41 |
| C.2.7 BENEFITS:..... | 62 |
| D.1 Introduction..... | 1 |
| D.2 PGS_AA_dcw..... | 2 |
| D.2.1 Data Sets Accessed..... | 2 |
| D.2.2 Outline Functionality..... | 4 |
| D.2.2.1 Outline..... | 4 |
| D.2.3 Optimal Operation..... | 5 |
| D.2.4 Upgrades..... | 5 |
| D.2.4.1 Access Speed..... | 5 |
| D.2.4.2 Additional Coverages..... | 5 |
| D.3 PGS_AA_dem, PGS_AA_2DRead, PGS_AA_2Dgeo, PGS_AA_3DRead, PGS_AA_3Dgeo..... | 5 |
| D.3.1 Data Sets Accessed..... | 5 |
| D.3.1.1 Introduction..... | 5 |
| D.3.1.2 Support and format files..... | 7 |
| D.3.1.2.1 Support File..... | 8 |
| D.3.1.2.2 Freeform data description..... | 9 |
| D.3.2 Functionality and Operation..... | 10 |
| D.3.2.1 Outline Functionality..... | 10 |
| D.3.2.2 Parameters and the indexFile..... | 10 |
| D.3.2.3 Use of User Specified and Auto-Operations..... | 12 |
| D.3.2.4 Operational Environment..... | 14 |

| | |
|--|----|
| D.3.3 Optimal Operation..... | 14 |
| D.3.3.1 Buffering | 14 |
| D.3.3.2 Multiple calls..... | 15 |
| D.3.3.3 Pre-processing, formats and file sizes | 15 |
| D.3.4 Setting up new/user data sets | 15 |
| D.3.5 Upgrades | 16 |
| D.3.5.1 Interaction with HDF files | 16 |
| D.3.5.2 Other format types for user files | 16 |
| D.3.5.3 New Operations..... | 16 |
| D.4 PGS_AA_PeVA..... | 16 |
| D.4.1 Data Sets accessed..... | 16 |
| D.4.2 Outline Functionality | 16 |
| D.4.3 Optimal Operation..... | 17 |
| D.4.4 Upgrades | 17 |
| E.1 Preparing Simulated CERES L0 Files..... | 1 |
| E.1.1 Sample Session..... | 2 |
| E.2 CERES Level 0 processing code using the SDP Toolkit | 4 |
| E.2.1 Notes:..... | 4 |
| F.1 Tropical Rainfall Measuring Mission (TRMM) File Formats..... | 1 |
| F.1.1 TRMM Files Schematic..... | 1 |
| F.1.2 Detached SFDU File..... | 2 |
| F.1.3 TRMM File Header | 2 |
| F.1.4 TRMM Packet Data..... | 3 |
| F.1.5 TRMM File Footer..... | 4 |
| F.2 EOS AM File Formats..... | 4 |
| F.2.1 EOS AM File Schematic..... | 4 |
| F.2.2 EOS AM File Header..... | 5 |
| F.2.3 EOS AM Packet Data | 5 |
| F.3 EOS PM File Formats..... | 6 |
| F.3.1 EOS PM File Schematic | 6 |
| F.3.2 EOS PM File Header | 6 |
| F.3.3 EOS PM Packet Data..... | 7 |

| | |
|---|----|
| F.4 ADEOS-II File Formats..... | 8 |
| F.4.1 ADEOS-II File Schematic | 8 |
| F.4.2 ADEOS-II File Header | 8 |
| F.4.3 ADEOS-II Packet Data..... | 8 |
| G.1 Projection Id's..... | 1 |
| G.1.1 NOTES..... | 1 |
| G.2 GCTP Error Messages..... | 2 |
| G.2.1 NOTES..... | 6 |
| G.3 UTM Zone Codes..... | 7 |
| J.1 Introduction | 1 |
| J.2 Development of the Core Metadata Model | 2 |
| J.3 ECS Granule Level Metadata..... | 3 |
| J.4 Satisfying Mandatory Requirement and Determining Specific Attributes..... | 4 |
| J.5 Metadata Toolkit Usage | 5 |
| J.6 Metadata Control File (MCF)..... | 7 |
| J.6.1 Purpose of the MCF..... | 7 |
| J.6.2 Structure of the MCF..... | 7 |
| J.6.2.1 GROUPS | 7 |
| J.6.2.2 OBJECTS | 8 |
| J.6.2.3 PARAMETERS..... | 10 |
| J.6.3 Internal Syntax..... | 13 |
| J.6.4 Constructing the MCF | 14 |
| J.6.4.1 Self Describing Attributes | 15 |
| J.6.4.2 Example header | 17 |
| J.6.5 Management of the MCF..... | 20 |
| J.6.5.1 Management of Master Groups..... | 20 |
| J.6.5.2 Management of Objects..... | 21 |
| J.6.5.3 Management of Parameters..... | 21 |
| J.6.6 Example of Output Header..... | 22 |
| J.6.7 Inclusion of Multiple Attribute Values..... | 24 |

J.7 Toolkit Utilization of the MCF.....28

- J.7.1 Overview28
- J.7.2 Scenario30

J.8 MCF Specification.....32

- L.1 Spacecraft Ephemeris File Format1
- L.2 Spacecraft Attitude File Format3
- L.3 Quality Flags.....5

J.1 Introduction

The purpose of this appendix is to provide detailed guidance to algorithm and PGE developers on the subject of granule level metadata population (i.e., metadata having different values in each product granule). Introductory comments put this class of metadata in context.

Within ECS, the term "metadata" relates to all information of a descriptive nature which is associated with the product or dataset . This includes such information as:

- data elements usually found in the product or file header
- documentation that accompanies the production algorithm software
- data origin information
- software used to create the data
- information used in the advertisement of the data

These types of information have been analyzed and developed into a core metadata model. Reference Document: DID311 ("Science Data Processing Segment Database Design and Database Schema Specification for the ECS Project, Volume 1: Central Design Artifacts. 311-CD-002-004 12/95").

J.2 Development of the Core Metadata Model

In relation to granule level metadata, the modules deemed most important within the metadata model are the Granule, Spatial and Temporal modules since they relate most closely to the inventory.

To avoid misunderstandings concerning the terms 'data set' and 'product', the concept of a collection of granules is used throughout ECS metadata related documentation. A collection is a logically organized set of related or grouped granules, chosen by the data provider. The granule is the lowest denominator in the equation may be assigned to more than one collection. It is generally construed as being the smallest aggregation of data that is independently managed i.e. an inventory record.

J.3 ECS Granule Level Metadata

The content of metadata, or the actual parameters which make up the core metadata and the product specific metadata are not at issue for this Users Guide. What is specified is

- the granule level metadata,
- how these metadata are provided to the granule header,
- how the Metadata tools assist this process,
- what the PGE has to do,

- what is the outcome of running the tools

i.e. the granule level metadata population process.

To establish the data required, the core metadata has been subsetted to derive granule metadata. The subset is a small part of the core. The exact list will vary with the product level but will include at a minimum a spatial and temporal measure plus other attributes specified in DID 311 Appendix B as mandatory. The spatial and temporal measures are themselves groups of attributes having options.

Product specific metadata is additional information added for the unique characteristics which can be product, collection or site specific. These are described as non-core or product specific attributes.

In section J.8 the syntax for attributes is specified.

J.4 Satisfying Mandatory Requirement and Determining Specific Attributes

Appendix B of DID 311 (311-CD-002-004 12/95), gives detailed information as to the level of granule Metadata needed to describe a product, as well as the multiplicity for certain attributes. This is laid out in BNF.

Appendix B specifies the metadata attributes which are mandatory for different categories of product managed by ECS. For detailed descriptions of these attributes refer to the data dictionary, section 6.4 of DID 311. The categories of data product in relation to the level of metadata support required are as follows:

Full level of metadata - required for products generated with EOSDIS

Intermediate level of metadata - required for products generated outside EOSDIS, but ingested and used within EOSDIS

Limited level of metadata - applies to all other data sets. We expect very few collections to fall into this category.

For a more detailed description of the above levels of metadata refer to Appendix B DID311. Each level of metadata is expanded to show the production rules related to that level, in BNF format.

J.5 Metadata Toolkit Usage

The metadata tools are designed to manage the granule metadata which are to be generated for each EOS product.

The reasoning behind the tools is:

- to standardize the granule metadata produced against minimum standards
- to assist the science software to produce metadata in the correct formats and syntax

- to collate metadata required by products but not necessarily (directly or efficiently) accessible to science algorithms
- the need to handle parsable metadata fields which may be added to or altered at other points in the system
- the need to interface with ECS products using HDF libraries
- the need to provide ingest capability for PGE's as well as output

A detailed working example of the metadata tools can be found in the metadata tool prologs in the User Guide. This example highlights the toolkit calling sequence usage, and reiterates the importance of using the tools in the proper sequence. Examples are given in both C and FORTRAN.

The metadata tools are a mandatory suite, to be used for the purposes of managing designated metadata attributes. By providing a suite of tools and a template, a standard set of metadata can be provided with each granule and this will allow a minimum search functionality to be carried out.

The Metadata Control File (MCF) assists the Instrument teams in supplying metadata attributes to products being generated. It is a template designed to contain metadata attributes and assist in the management of all and the acquisition of certain of these attributes.

The PGE derives and generates most of the attribute values which will populate the MCF, the metadata tools assist in this process. The figure below shows all the tools which may be utilized and their purpose.

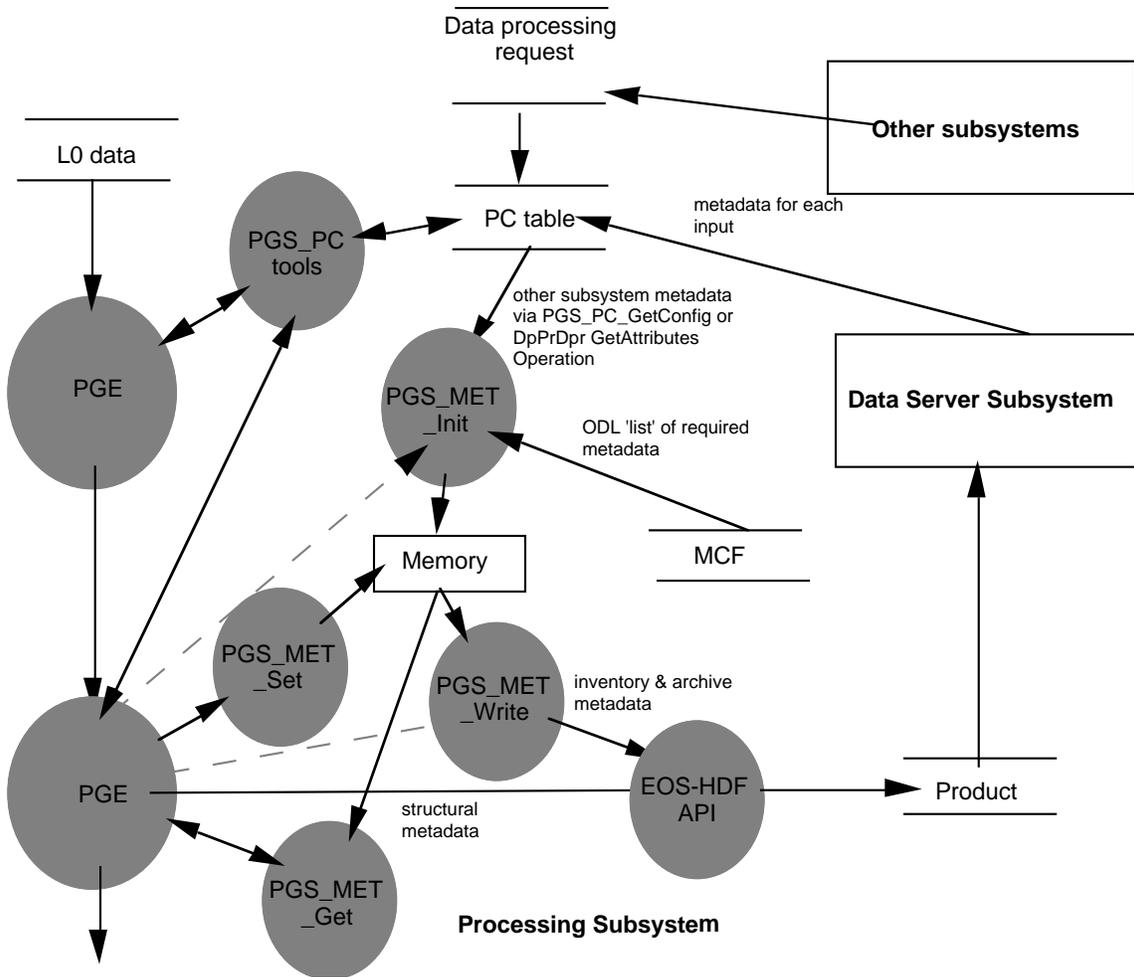


Figure J-1. Processing Subsystem

J.6 Metadata Control File (MCF)

The MCF assists the Instrument teams in supplying metadata attributes to product granules being generated. It is a template designed to contain metadata attributes "and assist in the management of all and the acquisition of certain of these attributes". The metadata to be attached to each product comes under two categories, Core and Product Specific. Core metadata will be included with every collection. Product specific will vary with each collection.

J.6.1 Purpose of the MCF

The MCF will be provided to every Instrument Team as a template—to be filled out and provided as part of the AI&T package so that the MCF can be staged alongside the product for access by the toolkit. It is the Instrument Teams role to perform the following:

- To liaise with the metadata development personnel in verifying the content of the granule specific inputs. To be familiar with Appendix B of DID 311. To specify the inputs which may be product specific and added to the MCF
- To use the MCF file from the latest Toolkit delivery and populate this template with product specific inputs
- To provide updates to the MCF when the need arises, or change the information when it resides on the data server (this capability will in future be handled by the Data Server, which will then create a new MCF.

Information which describes the setup of the actual files, SDS's, swaths etc. in the granule i.e. startrow, startcolumn are written and attached to the granule by the PGE using HDF-EOS calling sequences. This "structural" metadata is not used to populate the inventory, rather it is used to support the services which may be performed upon the granule. There is no direct association between the metadata groups set up in the MCF and the structural metadata. The MCF is not used to populate the structural metadata.

J.6.2 Structure of the MCF

The MCF is constructed of a number of major GROUPS, OBJECTS and PARAMETERS which describe the objects or metadata attributes. For an in depth description of the actual attributes which may be held within the MCF consult the data dictionary section of DID311.

J.6.2.1 GROUPS

A group contains a set of objects which all have a similar theme or tie. A group is also the means of attaching a set of attributes to a specific location.

The MCF consists of two or more "master groups", which split the data into that which will be copied and extracted into the inventory (one master group only) as well as being archived with the product, and that which will only be archived (one or more master groups). Only MASTERGROUP groups will ever be attached to an HDF attribute, using PGS_MET_Write.

Within these master groups, there are a number of subgroups. These have been included to contain sets of attributes which potentially may be repeated, or have multiple instances. These groups have names based on DID311 model class names, e.g. BoundingBox. When the user supplies their own product specific metadata to the MCF template, the attributes **must** be placed in one of the self describing groups; either 'AdditionalAttribute' or 'Parameter' (note only the latter is supported for release A). Values relating to these attributes are placed in the group 'InformationContent'.

Groups may be nested to allow for greater flexibility. In order to distinguish between a master group i.e. one that will be written to an HDF attribute, and a group within a master group, used to amalgamate similar objects, an attribute called GROUPTYPE is assigned the value MASTERGROUP. This allows the metadata tools a means of distinguishing between the two levels of group.

The MCF must start:

```
GROUP = INVENTORYMETADATA
```

```
    GROUPTYPE = MASTERGROUP
```

and end that master group:

```
END_GROUP = INVENTORYMETADATA
```

any nested groups within this master group must look like this:

```
    GROUP = EXAMPLENAME
```

```
    END_GROUP = EXAMPLENAME
```

J.6.2.2 OBJECTS

An object name relates directly to the attribute name in DID311 i.e. NorthBoundingBox. The template consists of a number of mandatory core objects, non mandatory core objects and product specific objects. An object must always follow the correct syntax:

```
OBJECT = EXAMPLENAME
```

```
END_OBJECT = EXAMPLENAME
```

Within this sequence are contained the parameters which describe this object .

Objects can also be nested, although only one level of nesting is possible within the metadata tools. Nesting is used, for a number of objects which may have two dimensional arrays as values i.e. GringPolygons. There may be a number of polygons, each having an exclusion flag and each polygon may be made up of many latitude and longitude measurements. In this scenario the container object is the representation of polygons. A new instance of the container object is created by the tools on output to the header whenever a new polygon is being described. The class attribute is used to define this new instance using PGS_MET_SetAttr. The objects class container within this container object would increment every time a new instance of the container object occurred.

e.g.

GROUP = GRing

OBJECT = GRingContainer

```
Data_Location= "NONE" /* necessary to i.d. a non-functional */  
                                /* container object */
```

```
CLASS = "M"
```

```
Mandatory = "TRUE"
```

```
OBJECT = ExclusionGRingFlag
```

```
    Data_Location= "PGE"
```

```
    CLASS = "M"
```

```
    TYPE = "STRING"
```

```
    NUM_VAL = 1
```

```
    Mandatory = "TRUE"
```

```
END_OBJECT = ExclusionGRingFlag
```

```
/* for each of the following objects there are at least 3 elements */
```

```
/* in each array */
```

```
OBJECT = GRingPointLatitude
```

```
    Data_Location = "PGE"
```

```
    CLASS = "M"
```

```
    TYPE = "DOUBLE"
```

```
    NUM_VAL = n                /* an array of max size n */
```

```
                                /* where n is at least 3*/
```

```
    Mandatory = "TRUE"
```

```
END_OBJECT = GRingPointLatitude
```

```
OBJECT = GRingPointLongitude
```

```
    Data_Location = "PGE"
```

```
    CLASS = "M"
```

```
    TYPE = "DOUBLE"
```

```
    NUM_VAL = n
```

```

        Mandatory = "TRUE"
    END_OBJECT = GRingPointLongitude
    OBJECT = GRingPointSequenceNo
        Data_Location = "PGE"
        CLASS = "M"
        TYPE = "STRING"
        NUM_VAL = n
        Mandatory = "TRUE"
    END_OBJECT = GRingPointSequenceNo
END_OBJECT = GRingContainer
END_GROUP = GRing

```

J.6.2.3 PARAMETERS

The metadata tools help to set metadata values for a product granule from three input sources—the Metadata Control File itself (MCF, pre-assigned values, entered during customization of the template by the instrument team), the Process Control File (PCF, which contains all file input and output specifications) and the PGE.

See example in J.6.2.2.

An object is described by a number of parameters, the parameters implemented within the MCF are: Data_Location, Mandatory, NUM_VAL, TYPE, CLASS and when Data_Location = MCF, an object has a Value assigned to it in the template. All objects must have Mandatory, TYPE, Data_Location, Num_Val present at all times. These fields are case insensitive in the current release of the Toolkit.

Data_Location - The value associated with an object in the template may be set via a number of different mechanisms. Data_Location indicates the source of population. Data_Location must be set for every object.

The attribute Data_Location may have the following values:

MCF - in this case, the object e.g. ShortName will have the value set whilst the template is being customized by the user/DAAC. ONLY when the Data_Location is equal to MCF does an object have a value assigned to it, not using any of the metadata toolkit routines. At present the template shows this example as value = xxxxx.

Values to be located in the MCF may be mandatory or non mandatory.

"MCF" **MUST** be in quotes as the value for Data_Location.

PGE - in this case, the object e.g. PointLatitude is set by the science software using PGS_MET_Set_Attr metadata tool. The majority of objects will be set this way.

Values to be supplied by the PGE may be mandatory or non mandatory.

"PGE" **MUST** be in quotes as the value for Data_Location.

PCF - in this case, the object will automatically be assigned a value during initialization of the MCF when using PGS_MET_Init. The value is derived from the runtime environment within the PCF. Metadata Toolkit code will locate the **object name** within the actual PCF delimiters. The name to be searched on must be written between the second and third delimiters in the PCF, and its corresponding value between the third and fourth delimiters. The object name and its value **MUST** exist in the PCF for this operation to succeed.

e.g. 12100|SATELLITE NAMES|("TRMM", "MODIS")

The string representing the parameter must be a unique Runtime parameter to search on. Any attribute names which are referenced in the PCF should be in **UPPER** case but **MUST** be unique, so that the metadata tools can match against the attribute name.

Values to be located in the PCF may be mandatory or non mandatory.

Quotes are only necessary when the datatype of the value is STRING and brackets are only required if NUM_VAL is greater than one.

e.g. 12101|MIN MAX DATA VALUES |(7, 99)

"PCF" **MUST** be in quotes as the value for Data_Location.

NONE - only necessary with nested objects, when the container object does not going to have a value associated with it. An example can be found in J.8 for GringPolygonContainer.

"NONE" **MUST** be in quotes as the value for Data_Location.

Mandatory - A mandatory statement must be set for every attribute. The mandatory attribute can have the values "TRUE" or "FALSE". Attributes in DID311 which have been specified as a mandatory metadata attribute for each granule, are flagged as "TRUE", as are any product specific attribute which will be inventoried. Where DID-311 specifies an attribute as Mandatory If Applicable, the mandatoriness should be decided on a case by case basis. This set of metadata should only have the value "FALSE" if it can not be applied to the data set, otherwise the value of mandatory should be set to "TRUE".

PGS_MET_Write returns an error if no value for a Mandatory attribute has been set, or the value not the correct Type. If the value is "FALSE" a warning will be returned for the same criterion. In both cases PGS_MET_Write completes its function setting the value to a

default, which can be found in the metadata tool prologs, for missing mandatory attributes and omitting the attribute where non-mandatory attributes have not been assigned a value. The Toolkit reports these errors and warnings in the log file.

For any metadata attribute whose value will be moved to the inventory for search purposes, the mandatory attribute must be set to "TRUE", and for any attributes to be archived with the granule, the value may be "TRUE" or "FALSE", depending on the importance of the attribute as determined by the Instrument Teams.

"TRUE" and "FALSE" must be enclosed in quotes in the MCF.

Type - The type attribute is attached to all objects in the MCF. The attribute allows the metadata tools to cast the data provided by the PGE to the correct data type. The values which may be utilized for this attribute are: "DATETIME", "INTEGER", "DOUBLE", "STRING" and "UNSIGNEDINT". Type must be present for every object. Note that since ODL does not support unsigned integers, the value written by the PGS_MET_Write tool may appear negative, but the Toolkit handles any conversion between signed and unsigned values based on the TYPE.

All values for Type must be enclosed in quotes.

CLASS - The attribute class is a mechanism for allowing attributes to occur multiple times, with the same name i.e. GRingPointLatitude. The permitted multiplicity for each attribute is described in DID 311, appendix B using BNF notation. Multiple values can occur in two forms, by a single dimensional array - SEE NUM_VAL, or an attribute can have two dimensions e.g. GRingPointLatitude, which has many points describing the latitude, and may have many Rings describing the GRing. Class is used to allow the repetition of the same attribute name, by assigning a different string value for each instance of class with the same attribute name. With each instance of the same attribute name, the PGS_MET_Set tool must be called with a CLASS suffix to the attribute name - giving the attribute a unique representation - known as multiplicity. The actual suffix used is of little consequence but integer increments are advised.

The CLASS system is set up in the MCF just once, using the CLASS = "M " statement, where M means multiple. By using this mechanism the MCF allows the user to specify how many attributes of the same name will have multiple instances, and the metadata tools, on encountering the CLASS = "M" statement will be able to handle this.

When using multiple objects and the CLASS attribute, the MCF must always follow these rules. Any object which is multiply defined is usually bounded by a general object e.g. GRingContainer, (introduced to describe complex metadata) and MUST then be bounded by a group name e.g. GRing. The CLASS for the general object must be set to "M".

The value which the user sets each instance of CLASS to, must be a unique string. CLASS is only present for objects and groups which have multiple instances.

If the value of CLASS is a character string it must be enclosed in quotes.

NUM_VAL - This attribute is used to describe a single dimensional array of values for an attribute. This value must be set to the maximum number of values held within the array. Any number of attributes up to this limit may be set. **NUM_VAL** must be present within every object.

Value - This attribute is only present in the MCF template when the Data_Location attribute is set to MCF. e.g. ShortName. Value is only set in the template for attributes which are seen as being 'static' in nature. Value is, however, set for all attributes when PGS_MET_Write tool creates the ODL header to be attached to the HDF-EOS attribute. As noted previously, if a value has not been filled by either the PGE, PCF or MCF then either a default value will be set or the attribute will not be written, and an error or warning will be returned from PGS_MET_Write.

J.6.3 Internal Syntax

ODL

The MCF template is closely based on Object Description Language (ODL) libraries. ODL accesses data which is kept in a hierarchical format using a GROUP, OBJECT and PARAMETER hierarchy. A group being an agglomeration of objects which are similar in some form, and which are described by a number of attributes.

ODL allows the nesting of both Objects and Groups. Both of these are nested in the MCF, and described in section J.6.2.

Any information pertinent to PGE developers about ODL, and its functionality is contained within this document, however should any additional information be sought, the URL's are

- WWW address <http://stardust.jpl.nasa.gov/stdref/chap12.htm>

A description of the ODL processing libraries.

- WWW address <http://joy.gsfc.nasa.gov/CCSDS-DocLib.html>

ODL uses a parameter = keyword statement to navigate attributes. Additional information on this notation can be found at the URL above.

When navigating through the MCF template, the ODL is translated internally into an ODL tree structure, enabling searches to be performed on Groups, Objects and Keywords. It is this means of identifying attributes, which enables the metadata tools to manage the metadata and assign values to attributes.

- ODL handles parameters and values in Upper case. The metadata toolkit converts all character strings to upper case, as an added protection and fail-safe device.
- ODL only recognizes a character string value when it is in quotation marks.
- ODL only accepts Time/Date in CCSDS A (UTC) Format.

- ODL only will accept INTEGER, UNSIGNEDINT, DOUBLE, DATETIME or STRING as a value for type

J.6.4 Constructing the MCF

Constructing the MCF is largely a matter of taking the template provided and editing it for the specific product or products for which it is being used. Editing must be in line with the rules for mandatory and non-mandatory metadata attributes laid down in Appendix B of DID311. However, instrument teams have a large amount of latitude and decision making power in setting many of the attributes.

Section J.6.4.1. below describes one of the more complex aspects of the MCF while J.6.4.2 provides an example output header for these types.

J.6.4.1 Self Describing Attributes

A number of the objects in the MCF are termed self describing. This term derived from DID311 data model is used to house attributes, which may not fit into a any class and therefore are described using name, type, description and sometimes value. The attributes are those relating to Parameter, AdditionalAttribute, SensorCharacteristics and PlatformCharacteristics and to a lesser extent VerticalSpatialDomain, Locality, and RegularPeriodic time.

Self describing values are described in the MCF using the following style - ParameterName, which has a corresponding ParameterValue for non-core, possibly product specific, attributes changing dynamically with each granule. The TYPE is Parameter DataType (i.e. user defined). Several of the self describing groups can be multiple; i.e. the group/object construct is 2-dimensional values sets; the first dimension is 'used' to contains single dimensional arrays (using NUM_VAL) while the second dimension contains the self describing attributes (CLASS = "M").

Some sensor characteristics change relatively rapidly and sometimes need to be stored in a database. The values vary according to data in the telemetry stream and are processed automatically by Science Software. They are set up in the same way as Parameter values. SensorCharacteristicValue has a datatype defined by SensorCharacteristictype.

e.g.

GROUP = InformationContent

OBJECT = InformationContentContainer

CLASS = "M" /* counter for each parameter */

Mandatory = "TRUE"

Data_Location = "NONE"

OBJECT = ParameterName

Data_Location = "PGE"

TYPE = "STRING"

```
        CLASS = "M"
        NUM_VAL = 1
        Mandatory = FALSE
    END_OBJECT = ParameterName
    OBJECT = ParameterValue
        Data_Location = "PGE"
        CLASS = "M"
        TYPE = "STRING"
        NUM_VAL = 5
        Mandatory = FALSE
    END_OBJECT = ParameterValue
END_OBJECT = InformationContentContainer
END_GROUP = InformationContent
GROUP = SensorCharacteristic
OBJECT = SensorCharacteristicContainer
    CLASS = "M"
    Mandatory = "TRUE"
    Data_Location = "NONE"
    OBJECT = SensorCharacteristicName
        Data_Location = "PGE"
        CLASS = "M"
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = SensorCharacteristicName
    OBJECT = SensorCharacteristicValue
        Data_Location = "PGE"
        CLASS = "M"
        NUM_VAL = 1
```

```

Mandatory = "FALSE"
TYPE = "STRING"
END_OBJECT = SensorCharacteristicValue
OBJECT = SensorShortName      /* The sensor and Instrument names are */
    Data_Location = "PGE"     /* provided to enable the inventory */
    CLASS = "M"               /* database to locate the correct */
    TYPE = "STRING"          /* characteristic name */
    NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = SensorShortName
OBJECT = InstrumentShortName
    Data_Location = "PGE"
    CLASS = "M"
    TYPE = "STRING"
    NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = InstrumentShortName
OBJECT = PlatformShortName
    Data_Location = "PGE"
    CLASS = "M"
    TYPE = "STRING"
    NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = PlatformShortName
END_OBJECT = SensorCharacteristicContainer
END_GROUP = SensorCharacteristic

```

J.6.4.2 Example header

An example of how self describing attributes would look in an actual product header:

```
GROUP = SensorCharacteristic
OBJECT = SensorCharacteristicContainer
  CLASS = "1"
  OBJECT = SensorCharacteristicName
    CLASS = "1"
    NUM_VAL = 1
    value = "tilt"
  END_OBJECT = SensorCharacteristicName
OBJECT = SensorCharacteristicValue
  CLASS = "1"
  NUM_VAL = 1
  value = 37.3
END_OBJECT = SensorCharacteristicValue
OBJECT = SensorShortName
  CLASS = "1"
  NUM_VAL = 1
  value = "scr"
END_OBJECT = SensorShortName
OBJECT = InstrumentShortName
  CLASS = "1"
  NUM_VAL = 1
  value = "MODIS"
END_OBJECT = InstrumentShortName
OBJECT = PlatformShortName
  CLASS = "1"
  NUM_VAL = 1
  value = "AM-1"
END_OBJECT = PlatformShortName
END_OBJECT = SensorCharacteristicContainer
```

```
OBJECT = SensorCharacteristicContainer
  CLASS = "2"
  OBJECT = SensorCharacteristicName
    CLASS = "2"
    NUM_VAL = 1
    value = "roll"
  END_OBJECT = SensorCharacteristicName
OBJECT = SensorCharacteristicValue
  CLASS = "2"
  NUM_VAL = 1
  value = 123.9
END_OBJECT = SensorCharacteristicValue
OBJECT = SensorShortName
  CLASS = "2"
  NUM_VAL = 1
  value = "scanning radiometer"
END_OBJECT = SensorShortName
OBJECT = InstrumentShortName
  CLASS = "2"
  NUM_VAL = 1
  value = "MODIS"
END_OBJECT = InstrumentShortName
OBJECT = PlatformShortName
  CLASS = "1"
  NUM_VAL = 1
  value = "AM-1"
END_OBJECT = PlatformShortName
END_OBJECT = SensorCharacteristicContainer
END_GROUP = SensorCharacteristic
```

J.6.5 Management of the MCF

The MCF will allow the user to distinguish between metadata which will be used to populate the inventory in the data server and used for inventory searches for that particular granule, as well as writing metadata which is important to the description of the granule, and needs to be kept with the granule as it is archived.

Within the MCF—the separate parts, e.g. Inventory and Archive metadata each become a mastergroup. The actual inputs within these groups are then assigned to Objects, with keywords representing the attribute of the object, i.e. value, type, data_location - each keyword will have a value. Translated into ODL nomenclature these three categories become nodes within the ODL tree, Groups become aggregation nodes—one per Group, or Object, Keywords become Parameter nodes, and the values assigned to them are attached to Value nodes.

The configuration of the MCF is such that any information which needs to be added at a later date e.g. QA data, can be slotted into the MCF without any change to the way the Toolkit routines function. This is an important aspect which prevents the need to alter and recompile code with each change to the MCF.

A full example MCF can be found in J.8. A file such as this is expected to be returned by all instrument teams, with the relevant granule specific metadata information completed, and any relevant product specific metadata added, which may be of use to other science, instrument teams or other users.

The granule metadata is structured in such a way that the instrument team may choose which attributes best describe the products they are providing for the objects relating to size, spatial and temporal measurements. As long as one method of description is chosen, the MCF will be complete, if however the instrument team believes more than one attribute is appropriate, then the MCF will accept this also.

Note:

The users of the metadata Toolkit routines must remember that any type casting they require will be set in the tool using ODL specific types. This does not interfere with the users own type casting of values returned from the Toolkit call.

J.6.5.1 Management of Master Groups

The master groups (INVENTORYMETADATA and ARCHIVEDMETADATA) are directed into different HDF attributes using the PGS_MET_Write tool. Inventory metadata is written to the HDF file attribute coremetadata, and archived metadata is written to archivemetadata. Note that there is no need to define structural metadata within an MCF. The structural metadata is automatically generated by the HDF-EOS APIs and has the attribute name "structmetadata.N" (N=0...9). Also note that internally, the PGS_MET_Write tools will create HDF attributes "coremetadata.N" (N=0...n) and archivemetadata.N, a new attribute being created whenever the attribute size exceeds HDF limitations. Thus there is no need for the MCF developer to worry about the predicted size of there MASTERGROUPS; it will be handled internally by the Toolkit.

INVENTORYMETADATA attributes apply to the whole granule and are parsed into the inventory for search and retrieve purposes (and **must** be written to an HDF attribute named "coremetadata"). This group is also archived with the product granule, and must include mandatory core metadata attributes and may include product specific attributes.

The group ARCHIVEDMETADATA includes the following ...

1. product specific attributes.
2. attributes specific to elements of the granule (e.g. grids, swaths or points).
3. core attributes not required by the inventory

This grouping is done for ease of management and viewing by later users. The ARCHIVEDMETADATA group must be written to an HDF attribute called "archivemetadata".

J.6.5.2 Management of Objects

All objects contained within a master group will be attached to the specified HDF attribute. It is not possible to attach only a nested group, held within the mastergroup, or the objects it contains.

J.6.5.3 Management of Parameters

Not all parameters held within the MCF are written to the product. The parameters which are written to describe the object are - Num_Val, Class and the value associated with the object. For an example of the metadata attached to the HDF product see section J.6.6.

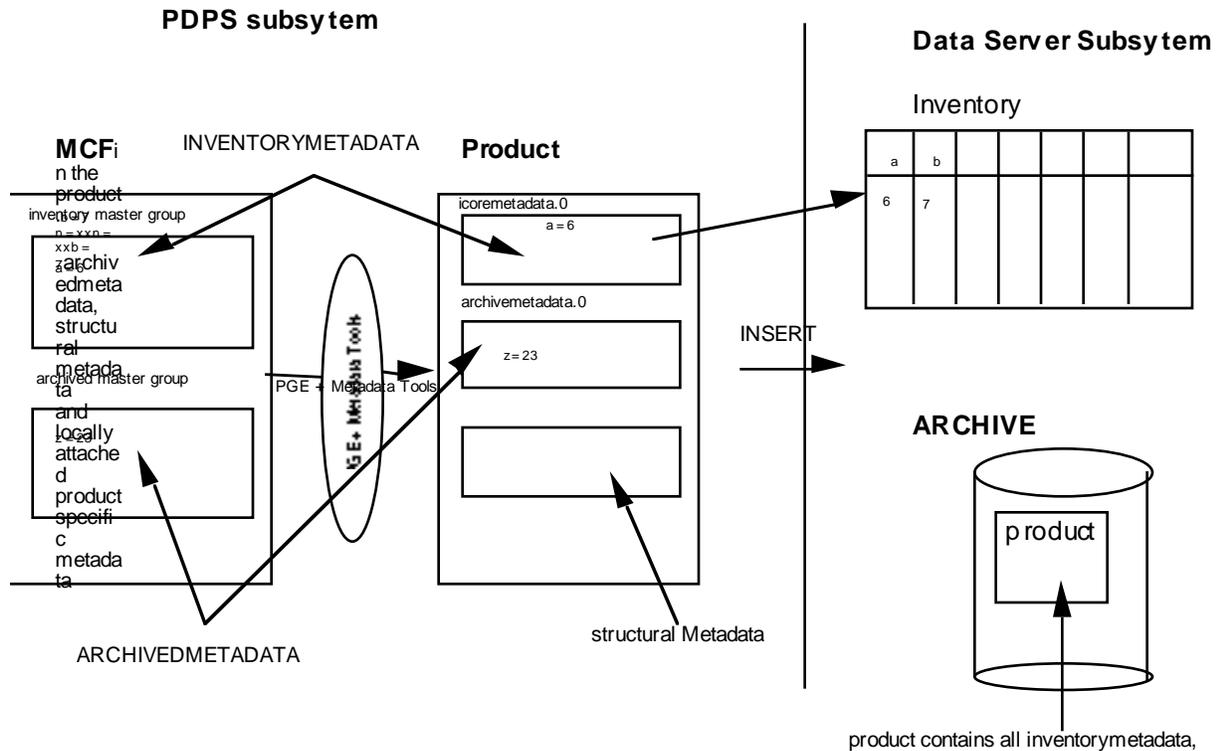


Figure J-2 Group Structure Parsing

J.6.6 Example of Output Header

An example of a small part of the Metadata which could be included within each HDF file as coremetadata is shown below. This shows the structure of the metadata written to the product, and what parameters are kept to describe the objects.

```

GROUP = INVENTORYMETADATA
    OBJECT = SHORTNAME
        NUM_VAL = 1
        VALUE = "MODIS"
    END_OBJECT = SHORTNAME
GROUP = GRING
    OBJECT = GRINGCONTAINER
        CLASS = 1
  
```

```

OBJECT = EXCLUSIONGRINGFLAG
    CLASS = 1
    NUM_VAL = 1
    VALUE = "N"
END_OBJECT = EXCLUSIONGRINGFLAG
OBJECT = GRINGPOINTLATITUDE
    CLASS = 1
    NUM_VAL = 3
    VALUE = (32.12, 41.1, 54.4)
END_OBJECT = GRINGPOINTLATITUDE
OBJECT = GRINGPOINTLONGITUDE
    CLASS = 1
    NUM_VAL = 3
    VALUE = (123.4, 156.8, 176.90)
END_OBJECT = GRINGPOINTLONGITUDE
OBJECT = GRINGPOINTSEQUENCENUMBER
    CLASS = 1
    NUM_VAL = 3
    VALUE = (1, 3, 2)
END_OBJECT = GRINGPOINTSEQUENCENUMBER
END_OBJECT = GRINGGONCONTAINER
OBJECT = GRINGGONCONTAINER
    CLASS = 2
    OBJECT = EXCLUSIONGRINGFLAG
        CLASS = 2
        NUM_VAL = 1
        VALUE = "Y"
    END_OBJECT = EXCLUSIONGRINGFLAG
    OBJECT = GRINGPOINTLATITUDE

```

```

        CLASS = 2
        NUM_VAL = 3
        VALUE = (22.12, 31.1, 44.4)
    END_OBJECT = GRINGPOINTLATITUDE
    OBJECT = GRINGPOINTLONGITUDE
        CLASS = 2
        NUM_VAL = 3
        VALUE = (23.4, 256.8, 276.90)
    END_OBJECT = GRINGPOINTLONGITUDE
    OBJECT = GRINGPOINTSEQUENCENUMBER
        CLASS = 2
        NUM_VAL = 3
        VALUE = (1, 2, 3)
    END_OBJECT = GRINGPOINTSEQUENCENUMBER
END_OBJECT = GRINGGONCONTAINER
END_GROUP = GRING
END_GROUP = INVENTORYMETADATA

```

J.6.7 Inclusion of Multiple Attribute Values

Attributes may have a single value associated with them, or an array of values. Denoting a single value is relatively simple in ODL. The values are enclosed within parenthesis e.g. if a single dimensional array of values were being set within the MCF - with `Data_Location = MCF`, the value would look like this (12, 34, 45, 88) or ("first", "second", "third").

To set a single dimensional array values using the metadata tools i.e. `Data_Location = PGE`, one call containing all the values in the array is performed using `PGS_MET_SetAttr`.

Two dimensional arrays, i.e. GringPolygons, where a polygon can not only have a number of values, but there may be a number of different polygons, utilize the array capability of ODL and `CLASS` (see section J.6.2.3). Each Gring would be denoted with a unique `CLASS` identifier, and any objects and attributes relating to that Gring would have the same unique `CLASS` identifier. Once the Gring was fully described, the unique `CLASS` identifier would be changed and the next Gring would be described. To keep track of the number of Grings within a granule container groups and objects are used.

To show the use of CLASS:- an example of the ODL tree structure derived from the MCF as it is being filled out memory. This is neither the MCF template nor the final header form.

GROUP = GRing

Object = GRingContainer

CLASS = 1

Mandatory = "TRUE"

OBJECT = ExclusionGRingFlag /* **RELATES TO GRING 1** */

Data_Location= "PGE"

CLASS = 1

TYPE = "STRING"

NUM_VAL = 1

Mandatory = "TRUE"

END_OBJECT = ExclusionGRingFlag

OBJECT = GRingPointLatitude /* **ARRAY DESCRIBING GRING 1** */

Data_Location = "PGE"

CLASS = 1

TYPE = "DOUBLE"

NUM_VAL = 6

Mandatory = "TRUE"

END_OBJECT = GRingPointLatitude

OBJECT = GRingPointLongitude /* **ARRAY DESCRIBING GRING 1** */

Data_Location = "PGE"

CLASS = 1

TYPE = "DOUBLE"

NUM_VAL = 6

Mandatory = "TRUE"

END_OBJECT = GRingPointLongitude

OBJECT = GRingPointSequenceNo

Data_Location = "PGE"

```

        CLASS = 1
        TYPE = "STRING"
        NUM_VAL = 6
        Mandatory = "TRUE"
    END_OBJECT = GRingPointSequenceNo
END_OBJECT = GRingContainer
OBJECT = GRingContainer
    CLASS = 2
    Mandatory = "TRUE"
    OBJECT = ExclusionGRingFlag /* RELATES TO GRING 2*/
        Data_Location= "PGE"
        CLASS = 2
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "TRUE"
    END_OBJECT = ExclusionGRingFlag
    OBJECT = GRingPointLatitude /* ARRAY DESCRIBING GRING 2 */
        Data_Location = "PGE"
        CLASS = 2
        TYPE = "DOUBLE"
        NUM_VAL = 6
        Mandatory = "TRUE"
    END_OBJECT = GRingPointLatitude
    OBJECT = GRingPointLongitude /* ARRAY DESCRIBING GRING 2 */
        Data_Location = "PGE"
        CLASS = 2
        TYPE = "DOUBLE"
        NUM_VAL = 6
        Mandatory = "TRUE"

```

```

END_OBJECT = GRingPointLongitude
OBJECT = GRingPointSequenceNo
    Data_Location = "PGE"
    CLASS = 2
    TYPE = "STRING"
    NUM_VAL = 6
    Mandatory = "TRUE"
END_OBJECT = GRingPointSequenceNo
END_OBJECT = GRingContainer
END_GROUP = GRing

```

J.7 Toolkit Utilization of the MCF

J.7.1 Overview

The prologs from the metadata tools code and are found in this Users guide along with relevant examples in both ANSI C and FORTRAN. Examples of the functions provided in both C and FORTRAN are present, as are notes which are specific to the functions. Actual calling sequence examples are provided in this Users Guide as well as a condensed version within the primer—accessible via a URL on the World Wide Web (WWW).

The calling sequences for the PGSTK metadata tools can be found in the prologs, which are found in the Toolkit Users guide above. In order to utilize the tools to their optimum capacity, they must be called in the specified sequence within algorithm code; i.e., PGS_MET_Init() (once for each physical MCF), then PGS_MET_SetAttr() (0-n times), then PGS_MET_Write (once for each HDF attribute). Note that with this release of the Toolkit, up to twenty MCF files may be opened and operated on at one time, with each call to PGE_MET_Init() must be given a unique handle. PGS_MET_GetSetAttr() and PGS_MET_GetPCAttr and PGS_MET_GetConfig can be called any number of times at any point after Init and before PGS_MET_Write.

IMPORTANT:

It is absolutely **imperative** that PGS_MET_Write be called to generate an ASCII version of the metadata in ODL format. In the current Toolkit release, this ASCII file will be generated automatically when the INVENTORYMETADATA section is written providing an HDF or HDF-EOS product is being generated. If an ASCII or simple binary product is being generated (e.g. an interim file) which requires archiving, it is necessary use PGS_MET_Write to generate this ASCII file.

The reason behind this is the method of populating the inventory during Insert of the product into the Data Server Subsystem. The insert service handles a number of different file formats

including HDF, and would require a number of different file opening mechanisms, by having the metadata in a flat ASCII file, one file opening mechanism is employed.

The insert service expects to insert the product and the ASCII file duplicating the metadata into the data server. The inventory will then be populated by opening up the ASCII file and copying the relevant attributes values into the inventory data base.

- The first step in reading from or writing to the MCF is with initialization. The contents of the MCF are read into memory, and any values which are to be set automatically from the PCF (i.e., where location = PCF) are located and inserted. The initialization process also checks to see if the MCF is in the correct ODL syntactical format.
- Once the MCF has been loaded into memory, values can be set against the relevant attributes names. The algorithm calling the tools has several methods available. The PGE can use PGS_MET_SetAttr to set values already known to the algorithm, or to set those values which are available from the Process Control File (PCF). If the algorithm needs to pick out an object value or a class of a named object from the MCF after it has been initialized, then PGS_MET_GetSetAttr is used.

There are two toolkit calls which enable the algorithm to extract metadata Information from the PCF.

- PGS_MET_GetPCAttr, which retrieves parameters which are either located as an HDF attribute in product files, or can be found in a separate ASCII file (the first occurrence of the attribute is returned from the file). These ASCII files **MUST** be in flat ODL format. The HDF attributes are guaranteed to be in this format if they have been written out to the file using the PGS_MET_Write function.
- PGS_MET_GetConfigData enables the user to obtain the configuration data parameters held within the process control table using an agreed label.

PGS_MET_GetConfigData will locate the value bar, for the parameter FOO in this PCF entry.

```
10255|FOO|"BAR"|/location/file|version
```

- Once the algorithm has finished retrieving and setting values in the memory, the final stage is to write the inventory and archive metadata values to the product. PGS_MET_Write writes the values out to an HDF file as an HDF 'attribute'. Note that a separate call to PGS_MET_Write is required for each master group.
- If the user needs to output the metadata to an ASCII (for non-HDF or non-HDF-EOS products), this is also possible, by calling PGS_MET_Write again with a variation in the calling sequence - see example in J.7.2. mdHandles[0] must be

supplied to `PGS_MET_Write` to output an ASCII file. The name and path of that ASCII file must be given in the PCF file. The file id used in this example is 10255.

Error messages, and an error return table and detailed descriptions of what they mean, can be found in the Users Guide prologs.

This sequence must be repeated for each MCF used by the PGE.

The format of the resulting metadata values written into the product (i.e., an HDF attribute) using HDF library calls is detailed in section J.6.6.

In order to further explain the utilization of the metadata toolkit routines, the following scenario shows examples and which tools would be utilized to perform them.

J.7.2 Scenario

Retrieval of Data from an HDF file used as input to current processing specified in the PCF, and setting that attribute in a new product.

STEP 1—Initialize MCF

PGS_MET_Init(filelogical, metadata handles)

STEP 2—Extract Value from file in PCF

PGS_MET_GetPCAttr(product file id, product version number, name of hdf attribute containing metadata, metadata parameter, returned metadata parameter value)

The output will be the value of the metadata attribute from the HDF metadata attribute or header. e.g., Obtain the `QA_Percentage_of_MissingData` from an input product.

STEP 3—Write the value extracted to the MCF in memory

PGS_MET_SetAttr(metadata group name, name.class of parameter, value to be inserted)

This will locate the group in the MCF, then the object name, and class if this is specified, and attach a new attribute to the object, which will hold the value to be associated with that attribute.

STEP 4—A value already held in the MCF in memory is needed to calculate a new value for a product specific object.

PGS_MET_GetSetAttr(metadata group name, name.class of parameter, value to be passed back)

STEP 5—In order to calculate this new value, information is also needed from the Configuration parameters set up in the Process Control File.

PGS_MET_GetConfigData(name of parameter, value to be passed back)

This will search the Process control file, and return the value back to the algorithm.

STEP 6—The PGE has used the two inputs to calculate a new value for one of the MCF objects, and wants to write it to the MCF held in memory.

PGS_MET_SetAttr(metadata group name, name.class of parameter, value to be inserted)

STEP 7—The PGE has finished setting all the values which are mandatory in the MCF, here is still some relevant granule information which the PGE wants to add to the MCF. The PGE accomplishes this by adding this information to the **PRODUCT_SPECIFIC_METADATA** group. Located within this group lies the object names the instrument team has already specified as being product specific.

PGS_MET_SetAttr(product specific metadata group name, name.class of parameter, value to be inserted)

STEP 8—After multiple calls to **PGS_MET_SetAttr** the MCF in memory is now complete, all the granule specific metadata has been set, the relevant product specific metadata has been set, the PGE now writes the metadata out as an HDF attribute attached to the product.

PGS_MET_Write(metadata group to be written out, HDF file attribute name ,HDF file ID)

STEP 9 —If the user wants to write out the MCF in memory as an ASCII file as well as attaching it to the HDF file, this is possible using **PGS_MET_Write** with a number of different inputs.

PGS_MET_Write(metadata group to be written out, HDF file attribute name [MUST BE SET TO NULL (char *)NULL],HDF file ID [MUST BE SET TO NULL (PGSt_Integer)NULL])

The user must give the `mdHandle[0]`, reserved to point to the whole MCF and the rest of the arguments as NULL. The name of the ASCII file must be defined in the Product Output Files section of the PCF with the logical Id 10255, in accordance with the defined rules for the PCF.

J.8 MCF Specification

This template can be used to develop individual MCF's

GROUP = INVENTORYMETADATA

GROUPTYPE = MASTERGROUP

/* *****Associate granule with collection ***** */

/* *****IMPORTANT - wherever xxxx or XXX or n or TBC is used */

/* within this MCF, these are values which the USER will */

/* ultimately fill in themselves - the MCF will not function with */

/* these original values!!!! ***** */

/* IMPORTANT - any text comment in the MCF MUST be enclosed */

/* per line with /* and */. Unlike C where a block of text may */

/* be enclosed */

/* All attributes EXCEPT those at the highest level associated */

/* with ECSDDataGranule (DID 311l) must be grouped together */

/* The groups MUST relate to the CLASS names within DID 311 */

/* Container Objects, within a group, must duplicate the group */

/* name and append Container */

/* IMPORTANT The following Groups are Self Describing */

/* Parameter, SensorCharacteristic, VerticalSpatialDomain */

/* PlatformCharacteristic, Locality, RegularPeriodic and */

/* AdditionalAttribute */

/* They all have values which are self typed - these values */

/* will be set up in the descriptor, prior to MCF generation */

/* For MCF purposes, the values of all of these attributes */

/* will be of type STRING. The database will retain the correct */

/* types. The PGE/user MUST convert the value they are */

/* assinging to type string, before calling PGS_MET_SET */

/* and setting the value of the attribute. */

/* The following metadata attributes are required for ALL classes of product */

```
OBJECT = ShortName          /* Name of attribute */
    Data_Location = "MCF"   /* Set in MCF */
    Value = "xxxx"         /* The value replaces xxxx */
    TYPE = "STRING"
    NUM_VAL = 1
    Mandatory = "TRUE"
```

END_OBJECT = ShortName

```
OBJECT = VersionID
    Data_Location = "MCF"
    Value = "xxxx"
    TYPE = "STRING"
    NUM_VAL = 1
    Mandatory = "TRUE"
```

END_OBJECT = VersionID

/* The size of the granule may be set once in the collection for */
/* all granule OR calculated routinely per granule (in which case */
/* the Data_location = PGE); which ever is most useful */

```
OBJECT = SizeMBECSDataGranule
    Data_Location = "PGE"
    TYPE = "INTEGER"
    NUM_VAL = 1
    Mandatory = "TRUE"          /* Optional for limited class */
```

END_OBJECT = SizeMBECSDataGranule

```
OBJECT = ReprocessingActual
    Data_Location = "MCF"
    Value = "xxxx"
    TYPE = "STRING"
    NUM_VAL = 1
```

```

        Mandatory = "TRUE"          /* Optional for limited and intermediate classes */
END_OBJECT = ReprocessingActual
OBJECT = ReprocessingPlanned
        Data_Location = "MCF"
        Value = "xxxx"
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "TRUE"          /* Optional for limited and intermediate classes */
END_OBJECT = ReprocessingPlanned
/* There are various ways of expressing spatial coverage. */
/* ECS requires that the following be supplied. */
/* BoundingBox (mandatory if applicable) */
/* plus none or more of... */
/* Point */
/* GPolygonContainer */
/* Circle */
/* OrbitCalculatedSpatialDomain */
/* Each of these is a compound of several attributes as specified */
/* below */
/* BoundingBox set; these must occur only once for each */
/* granule */
GROUP = BoundingBox
        OBJECT = EastBoundingBoxCoordinate
                Data_Location = "PGE"
                TYPE = "DOUBLE"
                NUM_VAL = 1
                Mandatory = "TRUE"    /* Mandatory if applicable for all classes*/
END_OBJECT = EastBoundingBoxCoordinate
OBJECT = WestBoundingBoxCoordinate

```

```

        Data_Location= "PGE"
        TYPE = "DOUBLE"
        NUM_VAL = 1
        Mandatory = "TRUE"          /* Mandatory if applicable for all classes*/
    END_OBJECT = WestBoundingCoordinate
    OBJECT = NorthBoundingCoordinate
        Data_Location = "PGE"
        TYPE = "DOUBLE"
        NUM_VAL = 1
        Mandatory = "TRUE"          /* Mandatory if applicable for all classes*/
    END_OBJECT = NorthBoundingCoordinate
    OBJECT = SouthBoundingCoordinate
        Data_Location = "PGE"
        TYPE = "DOUBLE"
        NUM_VAL = 1
        Mandatory = "TRUE"          /* Mandatory if applicable for all classes*/
    END_OBJECT = SouthBoundingCoordinate
END_GROUP = BoundingRectangle
/* there may be many GRingContainer object sets per */
/* GRing group i.e. many grings in each granule */
/* GRings must be sequenced i.e. sequence number in a clockwise */
/* manner, the area to the right of the direction of travel being the */
/* enclosed area */
GROUP = GPolygon
OBJECT = GRingContainer
    Data_Location= "NONE" /* necessary to i.d. a non-functional */
                        /* container object */
    CLASS = "M"
    Mandatory = "TRUE"

```

```

OBJECT = ExclusionGRingFlag
    Data_Location= "PGE"
    CLASS = "M"
    TYPE = "STRING"
    NUM_VAL = 1
    Mandatory = "FALSE" /* Mandatory if Grings are specified */
END_OBJECT = ExclusionGRingFlag

```

/* for each of the following objects there must be at least 3 elements */

/* in each array */

```

OBJECT = GRingPointLatitude
    Data_Location = "PGE"
    CLASS = "M"
    TYPE = "DOUBLE"
    NUM_VAL = 3 /* an array of min size 3 */
                /* can be any integer >= 3 */
    Mandatory = "FALSE" /* Mandatory if Grings are specified */
END_OBJECT = GRingPointLatitude

```

```

OBJECT = GRingPointLongitude
    Data_Location = "PGE"
    CLASS = "M"
    TYPE = "DOUBLE"
    NUM_VAL = 3 /* Same number as GRingPointLatitude */
    Mandatory = "FALSE" /* Mandatory if Grings are specified */
END_OBJECT = GRingPointLongitude

```

```

OBJECT = GRingPointSequenceNo
    Data_Location = "PGE"
    CLASS = "M"
    TYPE = "INTEGER"
    NUM_VAL = 3 /* Same number as GRingPointLatitude */

```

```

        Mandatory = "FALSE"  /* Mandatory if Grings are specified */
    END_OBJECT = GRingPointSequenceNo
END_OBJECT = GRingContainer
END_GROUP = GPolygon
/* granule may be described using one circle. */
/* If circle is used then all attributes within the group are mandatory */
GROUP = Circle
    OBJECT = CenterLatitude
        Data_Location = "PGE"
        TYPE = "DOUBLE"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = CenterLatitude
    OBJECT = CenterLongitude
        Data_Location = "PGE"
        TYPE = "DOUBLE"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = CenterLongitude
    OBJECT = RadiusValue  /* assumed to have the type */
                        /* of RadiusUnits */
        Data_Location = "PGE"
        TYPE = "DOUBLE"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = RadiusValue
    OBJECT = RadiusUnits
        Data_Location = "PGE"
        TYPE = "STRING"

```

```

        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = RadiusUnits
END_GROUP = Circle
/* granule may be described using single point.*/
/* If point is used then all attributes within point are mandatory */
GROUP = Point
    OBJECT = PointLatitude
        Data_Location = "PGE"
        TYPE = "DOUBLE"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = PointLatitude
    OBJECT = PointLongitude
        Data_Location = "PGE"
        TYPE = "DOUBLE"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = PointLongitude
END_GROUP = Point
/* when using Grid coordinate systems, ZoneIdentifier may, optionally, be specified */
GROUP = ZoneIdentifierClass
    OBJECT = ZoneIdentifier
        Data_Location = "PGE"
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = ZoneIdentifier
END_GROUP = ZoneIdentifierClass

```

/* when using locality text labels */

GROUP = GranuleLocality

OBJECT = LocalityValue

Data_Location = "PGE"

TYPE = "STRING"

NUM_VAL = 1

Mandatory = "FALSE" /* Optional for all classes */

END_OBJECT = LocalityValue

END_GROUP = GranuleLocality

/* Location information for vertical product data, at release A*/

/* only one vertical value per granule is expected. The TYPE is */

/* variable dependent on the VerticalSpatialDomainType, which should*/

/* be explicitly stated */

/* This group is optional for limited class, and mandatory if applicable for full and */

/* intermediate, but both attributes must be specified if used. */

GROUP = VerticalSpatialDomain

OBJECT = VerticalSpatialDomainValue

Data_Location = "PGE"

TYPE = "STRING"

NUM_VAL = 1

Mandatory = "FALSE"

END_OBJECT = VerticalSpatialDomainValue

OBJECT = VerticalSpatialDomainType

Data_Location = "PGE"

TYPE = "STRING"

NUM_VAL = 1

Mandatory = "FALSE"

END_OBJECT = VerticalSpatialDomainType

END_GROUP = VerticalSpatialDomain

/* The following group is optional for the limited class, and mandatory if */

/* applicable for intermediate and full classes*/

GROUP = OrbitCalculatedSpatialDomain

OBJECT = OrbitNumber

Data_Location = "PGE"

TYPE = "INTEGER"

NUM_VAL = 1

Mandatory = "FALSE" /* Mandatory if data is from a single orbit */

/* Otherwise should be deleted */

END_OBJECT = OrbitNumber

OBJECT = StartOrbitNumber

Data_Location = "PGE"

TYPE = "INTEGER"

NUM_VAL = 1

Mandatory = "FALSE" /* Mandatory if data is from >1 orbit */

/* Otherwise should be deleted */

END_OBJECT = StartOrbitNumber

OBJECT = StopOrbitNumber

Data_Location = "PGE"

TYPE = "INTEGER"

NUM_VAL = 1

Mandatory = "FALSE" /* Mandatory if data is from >1 orbit */

/* Otherwise should be deleted */

END_OBJECT = StopOrbitNumber

OBJECT = LongitudeOfEquatorCrossing

Data_Location = "PGE"

TYPE = "DOUBLE"

NUM_VAL = 1

Mandatory = "FALSE" /* Mandatory if this GROUP exists */

```

END_OBJECT = LongitudeOfEquatorCrossing
OBJECT = DateTimeOfEquatorCrossing
    Data_Location = "PGE"
    TYPE = "DATETIME"
    NUM_VAL = 1
    Mandatory = "FALSE"      /* Mandatory if this GROUP exists */
END_OBJECT = DateTimeOfEquatorCrossing
END_GROUP = OrbitCalculatedSpatialDomain
/* There are two ways to express temporal extent at the */
/* granule level. These are RangeDateTime or SingleDateTime */
/* each of which is a compound. It is MANDATORY to supply ONE of these */
/* for intermediate and full products, and optional to supply ONE for limited class */
/* products. Whichever is used, the other should be deleted from this template */
/* Repeating sets of time measures can be placed in the ARCHIVEDMETADATA */
/* master group*/
/* range time */
GROUP = RangeDateTime
    OBJECT = RangeBeginningDateTime
        /* Format is YYYY-MM-DDTHH:MM:SS.SSSS....Z or */
        /* YYYY-DDDTHH:MM:SS.SSSS....Z */
    Data_Location = "PGE"
    TYPE = "DATETIME"
    NUM_VAL = 1
    Mandatory = "FALSE"
END_OBJECT = RangeBeginningDateTime
OBJECT = RangeEndingDateTime
    /* Format is YYYY-MM-DDTHH:MM:SS.SSSS....Z or */
    /* YYYY-DDDTHH:MM:SS.SSSS....Z */
    Data_Location = "PGE"

```

```

        TYPE = "DATETIME"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = RangeEndingDateTime
END_GROUP = RangeDateTime
/* SingleDateTime */
GROUP = SingleDateTime
    OBJECT = CalendarDateTime
        /* Format is YYYY-MM-DDTHH:MM:SS.SSSS....Z or */
        /* YYYY-DDDTHH:MM:SS.SSSS....Z */
        Data_Location = "PGE"
        TYPE = "DATETIME"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = CalendarDateTime
END_GROUP = SingleDateTime
/* Information for data quality rating. */
/* each of the following QA - quality flags or values may be set */
/* for the whole granule and-or for science parameters within */
/* each granule. Only whole granule measures for inventory */
/* purposes are included in this master group, repeating sets */
/* can be placed into the ARCHIVEDMETADATA group using */
/* arrays and if necessary nested group and object and CLASS = "M" */
/* statements where 2 dimensional sets of values per attribute */
/* are required */
/* This group is optional for limited and intermediate class products */
/* and mandatory for full class products. If used, then the percentage of */
/* out of bounds data MUST be supplied, the others being optional */
GROUP = QAStats

```

```

OBJECT = QAPercentInterpolatedData
    Data_Location = "PGE"
    TYPE = "INTEGER"
    NUM_VAL = 1
    Mandatory = "FALSE"
END_OBJECT = QAPercentInterpolatedData
OBJECT = QAPercentOutOfBoundsData
    Data_Location = "PGE"
    TYPE = "INTEGER"
    NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = QAPercentOutOfBoundsData
OBJECT = QAPercentMissingData
    Data_Location = "PGE"
    TYPE = "INTEGER"
    NUM_VAL = 1
    Mandatory = "FALSE"
END_OBJECT = QAPercentMissingData
END_GROUP = QAStats
/* The following group is optional for limited and intermediate class products */
/* but mandatory for full. Only one of the flags could be set, normally the Automatic */
/* If used, both the flag and explanation must be assigned a value */
GROUP = QACollectionStats
    OBJECT = AutomaticQualityFlag
        Data_Location = "PGE"
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = AutomaticQualityFlag

```

```

OBJECT = OperationalQualityFlag
    Data_Location = "PGE"
    TYPE = "STRING"
    NUM_VAL = 1
    Mandatory = "FALSE"
END_OBJECT = OperationalQualityFlag
OBJECT = ScienceQualityFlag
    Data_Location = "PGE"
    TYPE = "STRING"
    NUM_VAL = 1
    Mandatory = "FALSE"
END_OBJECT = ScienceQualityFlag
OBJECT = QualityFlagExplanation
    Data_Location = "PGE"
    TYPE = "STRING"
    NUM_VAL = 1
    Mandatory = "FALSE"
END_OBJECT = QualityFlagExplanation
END_GROUP = QACollectionStats
/* these are URs made available to the PGE from the PCF */
/* which must be extracted by the PGE individually */
/* This group is optional for limited and intermediate products */
/* and mandatory for full class products*/
GROUP = InputGranule
OBJECT = InputPointer
    Data_Location = "PGE"
    TYPE = "STRING"
    NUM_VAL = 1    /* Alter as appropriate depending on */
                  /* number of inputs to product */

```

```

        Mandatory = "FALSE"
    END_OBJECT = InputPointer
END_GROUP = InputGranule
/* The following group is optional for limited and intermediate products */
/* and mandatory if applicable for full class products */
GROUP = AncillaryInputGranule
    OBJECT = AncillaryInputPointer
        Data_Location = "PGE"
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = AncillaryInputPointer
END_GROUP = AncillaryInputGranule
/* The following group is optional for limited products */
/* and mandatory if applicable for full and intermediate products */
GROUP = OrbitParametersGranule
    OBJECT = OrbitParametersPointer
        Data_Location = "PGE"
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = OrbitParametersPointer
END_GROUP = OrbitParametersGranule
/* The following group is optional for all classes of product */
GROUP = InformationContent
    OBJECT = InformationContentContainer
    CLASS = "M"          /* counter for each additional */
                        /* attribute */
    Mandatory = "TRUE"

```

```
Data_Location = "NONE"
/* EITHER AdditionalAttributeName or ParameterName is used */
/* to describe a ParameterValue if a value is appropriate. Note for release A */
/* ONLY ParameterName is supported, AdditionalAttribute is not. */
/* For situations where geophysical parameter */
/* content varies with the granule. The TYPE is ParameterDataType */
/* (i.e. user defined). This group object construct */
/* allows for 2-dimensional values sets; i.e. where each */
/* parameter has many values per granule. */
```

```
OBJECT = ParameterName
```

```
Data_Location = "PGE"
```

```
TYPE = "STRING"
```

```
CLASS = "M"
```

```
NUM_VAL = 1
```

```
Mandatory = "FALSE"
```

```
END_OBJECT = ParameterName
```

```
/* for non-core attributes which are not geophysical parameter */
/* content. The TYPE is */
/* AdditionalAttribute DataType (i.e. user defined). */
/* This group object construct */
/* allows for 2-dimensional values sets; i.e. where each */
/* additional attribute has many values per granule. */
/* This will not be supported at Release-A but is left here for */
/* later use. */
/* The value of additional attribute is ParameterValue */
```

```
OBJECT = AdditionalAttributeName
```

```
Data_Location = "PGE"
```

```
TYPE = "STRING"
```

```
CLASS = "M"
```

```

        NUM_VAL = 1
        Mandatory = "FALSE"

    END_OBJECT = AdditionalAttributeName
/* value may or may not be appropriate to describe the */
/* AdditionalAttributeName or ParameterName */
    OBJECT = ParameterValue
        Data_Location = "PGE"
        CLASS = "M"
        TYPE = "STRING"
        NUM_VAL = 1      /* Or greater depending on individual needs */
        Mandatory = "FALSE"

    END_OBJECT = ParameterValue
END_OBJECT = InformationContentContainer
END_GROUP = InformationContent
/* this next group used to track changes in parameter content */
/* (not parameter values); i.e. if science content varies per granule. */
/* Such variation presumes that a multitype collection has been */
/* established allowing for inventory level changes of this type. */
/* This is not supported at Release A */
GROUP = Parameter
    OBJECT = ParameterName
        Data_Location = "PGE"
        TYPE = "STRING"
        CLASS = "M"
        NUM_VAL = 1      /* Or greater depending on individual needs */
        Mandatory = FALSE

    END_OBJECT = ParameterName
END_GROUP = Parameter
/* these sensor characteristics change relatively rapidly and */

```

```
/* need to be store in a database. The values vary according */  
/* to data in the telemetry stream and are processed */  
/* automatically. They are set up in the same way as */  
/* Parameter values. SensorCharacteristicValue has */  
/* type SensorCharacteristictype. This group/object construct */  
/* allows for 2-dimensional values sets; i.e. where each */  
/* SensorCharacteristicValue has many values per granule. */  
/* This group is optional for all classes of data. */
```

```
GROUP = SensorCharacteristic
```

```
OBJECT = SensorCharacteristicContainer
```

```
CLASS = "M" /* allows for many sensorcharacteristics */  
/* per granule */
```

```
Mandatory = "TRUE"
```

```
Data_Location = "NONE"
```

```
OBJECT = SensorCharacteristicName
```

```
Data_Location = "PGE"
```

```
CLASS = "M"
```

```
TYPE = "STRING"
```

```
NUM_VAL = 1
```

```
Mandatory = "TRUE"
```

```
END_OBJECT = SensorCharacteristicName
```

```
OBJECT = SensorCharacteristicValue
```

```
Data_Location = "PGE"
```

```
CLASS = "M"
```

```
NUM_VAL = 1
```

```
Mandatory = "FALSE"
```

```
TYPE = "STRING"
```

```
END_OBJECT = SensorCharacteristicValue
```

```
OBJECT = SensorCharacteristicUnits
```

```

    Data_Location = "PGE"
    CLASS = "M"
    NUM_VAL = 1
    Mandatory = "FALSE"
    TYPE = "STRING"
END_OBJECT = SensorCharacteristicUnits
OBJECT = SensorCharacteristicType
    Data_Location = "PGE"
    CLASS = "M"
    NUM_VAL = 1
    Mandatory = "FALSE"
    TYPE = "STRING"
END_OBJECT = SensorCharacteristicType
OBJECT = SensorShortName
    Data_Location = "PGE"
    CLASS = "M"
    TYPE = "STRING"
    NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = SensorShortName
OBJECT = InstrumentShortName
    Data_Location = "PGE"
    CLASS = "M"
    TYPE = "STRING"
    NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = InstrumentShortName
OBJECT = PlatformShortName
    Data_Location = "PGE"

```

```

        CLASS = "M"
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "TRUE"
    END_OBJECT = PlatformShortName
/* Platform, Instrument and Sensor names needed to allow the */
/* database to locate the correct characteristic name */
END_OBJECT = SensorCharacteristicContainer
END_GROUP = SensorCharacteristic
/* this used to monitor instrument mode changes needed */
/* at granule level */
/* This is optional for all classes of data */
GROUP = OperationModeClass
    OBJECT = OperationMode
        Data_Location = "PGE"
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = OperationMode
END_GROUP = OperationModeClass
/* The following group is optional for all classes */
GROUP = Review
    OBJECT = ScienceReviewDate
        Data_Location = "MCF"
        TYPE = "DATETIME"
        NUM_VAL = 5      /* Any number can be assigned here */
        Mandatory = "FALSE"
    END_OBJECT = ScienceReviewDate
    OBJECT = ScienceReviewStatus

```

```

        Data_Location = "MCF"
        TYPE = "INTEGER"
        NUM_VAL = 5      /* Any number can be assigned here */
        Mandatory = "FALSE"
    END_OBJECT = ScienceReviewStatus
    OBJECT = FutureReviewDate
        Data_Location = "MCF"
        TYPE = "DATETIME"
        NUM_VAL = 5      /* Any number can be assigned here */
        Mandatory = "FALSE"
    END_OBJECT = FutureReviewDate
END_GROUP = Review
/* end of master group */
END_GROUP = INVENTORYMETADATA
GROUP = ARCHIVEDMETADATA
GROUPTYPE = MASTERGROUP
/* this master group may contain any core attributes group */
/* plus product specific attributes. Both may be single */
/* attributes or repeating. In the latter case, arrays plus the */
/* Group, Object and CLASS = "M" construct shown above should*/
/* be used. */
/* This group should be written using MET_WRITE. */
/* These attributes are NOT parsed into the inventory and */
/* are therefore NOT SEARCHABLE */
/* the following are core attributes not required to be searched */
/* but which may be mandatory or just useful - according to */
/* choices made in appendix B. */
/* these attribute needed in browse granules */

```

```

/* Shortname + same as in granule */
/ * BrowseDescription + */
/ * BrowseSize + */
/* SpatialDomainContainer + same as in granule */

```

```

/* [RangeDateTime | SingleDateTime] + same as in granule */

```

```

GROUP = Browse

```

```

OBJECT = BrowseSize

```

```

Data_Location = "PGE" /* may also be MCF (static) provided */
/* if not prone to change */

```

```

TYPE = "DOUBLE"

```

```

NUM_VAL = 1

```

```

Mandatory = "FALSE"

```

```

END_OBJECT = BrowseSize

```

```

OBJECT = BrowseDescription

```

```

Data_Location = "PGE"

```

```

NUM_VAL = 1

```

```

TYPE = "STRING"

```

```

Mandatory = "FALSE"

```

```

END_OBJECT = BrowseDescription

```

```

END_GROUP = Browse

```

```

/* the following are core attributes found in the */

```

```

/* INVENTORYMETADATA master group but duplicated in this */

```

```

/* master group in order to contain additional attribute values */

```

```

/* required to be in the product granule but not searchable; for */

```

```

/* example, quality measures related to individual geophysical */

```

```

/* parameters. The 2-dimensional formulation using arrays */

```

```

/* (NUM_VALS) and CLASS = "M" is used to accomodate many */

```

```

/* values in a self describing situation */

```

```
GROUP = SensorCharacteristic
OBJECT = SensorCharacteristicContainer
    CLASS = "M"
    Data_Location = "NONE"
    Mandatory = "FALSE"
    OBJECT = SensorCharacteristicName
        Data_Location = "PGE"
        CLASS = "M"
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = SensorCharacteristicName
    OBJECT = SensorCharacteristicValue
        Data_Location = "PGE"
        CLASS = "M"
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = SensorCharacteristicValue
    OBJECT = SensorShortName
        Data_Location = "PGE"
        CLASS = "M"
        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = SensorShortName
    OBJECT = InstrumentShortName
        Data_Location = "PGE"
        CLASS = "M"
```

```

        TYPE = "STRING"
        NUM_VAL = 1
        Mandatory = "FALSE"

    END_OBJECT = InstrumentShortName
END_OBJECT = SensorCharacteristicContainer
END_GROUP = SensorCharacteristic
GROUP = InformationContent

    OBJECT = InformationContentContainer

    CLASS = "M"          /* counter for each additional */
                        /* attribute */

    Mandatory = "TRUE"

    Data_Location = "NONE"

/* EITHER AdditionalAttributeName or ParameterName is used */
/* to describe a ParameterValue if a value is appropriate */
/* for situations where geophysical parameter */
/* content varies with the granule. The TYPE is Parameter */
/* DataType (i.e. user defined). This group object construct */
/* allows for 2-dimensional values sets; i.e. where each */
/* parameter has many values per granule. */

    OBJECT = ParameterName

        Data_Location = "PGE"

        TYPE = "STRING"

        CLASS = "M"

        NUM_VAL = 1

        Mandatory = "FALSE"

    END_OBJECT = ParameterName

/* for non-core attributes which are not geophysical parameter */
/* content. The TYPE is */
/* AdditionalAttribute DataType (i.e. user defined). */

```

```

/* This group object construct */
/* allows for 2-dimensional values sets; i.e. where each */
/* additional attribute has many values per granule. */
/* The value of additional attribute is ParameterValue */
    OBJECT = AdditionalAttributeName
        Data_Location = "PGE"
        TYPE = "STRING"
        CLASS = "M"
        NUM_VAL = 1
        Mandatory = "FALSE"
    END_OBJECT = AdditionalAttributeName
/* value may or may not be appropriate to describe the */
/* AdditionalAttributeName or ParameterName */
    OBJECT = ParameterValue
        Data_Location = "PGE"
        CLASS = "M"
        TYPE = "STRING"
        NUM_VAL = 3    /* Any integer number, depending on */
                     /* individual application */
        Mandatory = "FALSE"
    END_OBJECT = ParameterValue
END_OBJECT = InformationContentContainer
END_GROUP = InformationContent
/* Information for data quality rating. */
/* each of the following QA - quality flags or values may be set */
/* for the whole granule and-or for science parameters within */
/* each granule. */
GROUP = QAStats
    OBJECT = QAStatsContainer

```

```

CLASS = "M"          /* counter for each parameter */
Data_Location = "NONE"
Mandatory = "FALSE"
OBJECT = QAPercentInterpolatedData
    Data_Location = "PGE"
    CLASS = "M"
    TYPE = "INTEGER"
    NUM_VAL = 1      /* Any integer number, depending on */
                    /* individual application */
    Mandatory = "FALSE"
END_OBJECT = QAPercentInterpolatedData
OBJECT = QAPercentOutOfBoundsData
    Data_Location = "PGE"
    CLASS = "M"
    TYPE = "INTEGER"
    NUM_VAL = 1      /* Any integer number, depending on */
                    /* individual application */
    Mandatory = "FALSE"
END_OBJECT = QAPercentOutOfBoundsData
OBJECT = QAPercentMissingData
    Data_Location = "PGE"
    CLASS = "M"
    TYPE = "INTEGER"
    NUM_VAL = 1      /* Any integer number, depending on */
                    /* individual application */
    Mandatory = "FALSE"
END_OBJECT = QAPercentMissingData
/* parameter name may be required to link the QA measures to */
/* a particular parameter in the granule */

```

```

OBJECT = ParameterName
    Data_Location = "PGE"
    CLASS = "M"
    TYPE = "STRING"
    NUM_VAL = 1      /* Any integer number, depending on */
                    /* individual application */

    Mandatory = "FALSE"
END_OBJECT = ParameterName
END_OBJECT = QAStatsContainer
END_GROUP = QAStats
/* use NUM_VAL here to set multiple values of quality flag for */
/* components of the granule */
GROUP = QACollectionStats
    OBJECT = AutomaticQualityFlag
        Data_Location = "PGE"
        TYPE = "STRING"
        NUM_VAL = 1      /* Any integer number, depending on */
                        /* individual application */

        Mandatory = TBC
    END_OBJECT = AutomaticQualityFlag
END_GROUP = QACollectionStats
END_GROUP = ARCHIVEDMETADATA
END

```

Appendix K. POSIX Systems Calls Usage Policy

This appendix outlines the usage policy for the set of POSIX system API calls as outlined in:

IEEE Std 1003.1: POSIX Part 1: System Application Program Interface (API) [C Language]

IEEE Std 1003.9: POSIX FORTRAN77 Language Interfaces, Part 1: Binding for System Application Program Interface [API]

In general, the policy attempts to guard access to routines that may impact the SDPS where system resource management is an issue. This will be accomplished by restricting access to the standard POSIX system calls, as described below. The complete set of routines is listed in the "Identifier Index" of *IEEE Std 1003.1*, and in the body of *IEEE Std 1003.9*.

Table C–1 provides general policy "guidelines" for various classes of system routines. These guidelines are then used in determining the appropriate disposition for each of the POSIX system call functions on an individual basis. The general policy guidelines include:

- **Toolkit**—The described functionality is either accessible to the user via a "shadowing" routine in the PGS Toolkit, or it is used internally by the Toolkit itself. The Toolkit routine may be a simple subroutine call (or macro) wrapper around the "shadowed" function, or it may provide additional functionality that may be useful to the SDPS in accomplishing its resource management objectives. Direct calls to the respective POSIX API calls are prohibited within science algorithm code.
- **Prohibited**—Access to the described functionality is prohibited. Direct calls to the respective POSIX API calls are prohibited within science algorithm code.
- **Allowed**—Access to the described functionality is allowed through the standard POSIX API calls. The Toolkit itself makes calls to these routines in addition to those listed in the Toolkit category.

The algorithm integration and test facility will include "code checkers" to screen science algorithms for adherence. These code checkers will be provided as part of the PGS Toolkit to support the development of policy compliant algorithms. This should greatly facilitate the algorithm integration and test procedure.

Table K-1. POSIX Call Guidelines By Class

| Class | Description | Policy Guideline |
|---------------------|--|------------------|
| Process control | Process creation and termination; interprocess signaling and synchronization | Toolkit |
| Memory | Memory allocation, deallocation, and mapping | Toolkit |
| File I/O | File I/O routines; directory manipulation routines | Toolkit |
| Stream I/O | Stream I/O routines | Toolkit |
| Error / environment | Error recording and reporting; environment access | Toolkit |
| Ownership | Process ownership and groups; file ownership, permissions, and creation/access times | Prohibited |
| Miscellaneous | Math, "is...", "str...", and time functions | Allowed |
| Terminal I/O | Terminal I/O and characteristics | Prohibited |
| Status | System and resource status (read only) | Allowed |

Tables K-2 through K-10 constitute a listing of the entire set of POSIX C API calls, organized by class and policy as described above. Table K-11 provides a listing of the FORTRAN77 specific language library calls that do not have C API counterparts. Entries in **bold** indicate that a Toolkit "shadow" function has been created to perform this functionality.

Table K-2. POSIX Calls: Process Control

| Toolkit Routines | Prohibited Routines | Allowed Routines |
|--|---|------------------|
| exec...() _exit() fork() sig...() sleep() wait() waitpid() | abort() alarm(), PXFALARM() PXFEXEC...() PXFEXIT() PXFFASTEXIT() PXFFORK() kill(), PXFKILL() pause(), PXFPAUSE() PXFSIG...() PXFSLEEP() PXFWAIT() PXFWAITPID() | exit() |

Table K-3. POSIX Calls: Memory

| Toolkit Routines | Prohibited Routines | Allowed Routines |
|---|---------------------|---|
| calloc() free() malloc() realloc() | | calloc() free() malloc() realloc() |

Table K–4. POSIX Calls: File I/O

| Toolkit Routines | Prohibited Routines | Allowed Routines |
|---|---|------------------|
| access() close() creat() dup() dup2() lseek() open() pipe() read() remove() rename() tmpfile() tmpnam() write() | PXFACCESS() PXFCLOSE() PXFCREAT() PXFDUP() PXFDUP2() PXFLSEEK() PXFOPEN() PXFPIPE() PXFREAD() chdir(), PXFCHDIR() PXFRENAME() closedir(), PXFCLOSEDIR() fpathconf(), PXFFPATHCONF() getcwd(), PXFGETCWD() PXFWRITE() link(), PXFLINK() mkdir(), PXFMKDIR() mkfifo(), PXFMKFIFO() opendir(), PXFOPENDIR() pathconf(), PXFPATHCONF() readdir(), PXFREADDIR() rewinddir(), PXFREWINDDIR() rmdir(), PXRMDIR() unlink(), PXFUNLINK() utime() | |

Table K–5. POSIX Calls: Stream I/O

| Toolkit Routines | Prohibited Routines | Allowed Routines |
|--|--|--|
| fclose() fcntl(), fdopen() fileno() fopen() freopen() | setbuf() PXFFCNTL() PXFFDOPEN() PXFFILENO() clearerr() PXFPOSIXIO() | feof() ferrord() fflush(), PXFFLUSH() fgetc(), PXFFGETC() fgets() fprintf() fputc(), PXFFPUTC() fputs() fread() fscanf() fseek(), PXFFSEEK() ftell(), PXFFTELL() fwrite() getc(), PXFGETC() putc(), PXFPUTC() sprintf() sscanf() ungetc() |

Table K-6. POSIX Calls: Error/environment

| Toolkit Routines | Prohibited Routines | Allowed Routines |
|------------------|---------------------|---|
| | | assert() getenv(), PXFGETENV() perror() IPXFARGC() PXCLEARENV() PXFGETARG() PXFSETENV() |

Table K-7. POSIX Calls: Ownership

| Toolkit Routines | Prohibited Routines | Allowed Routines |
|-----------------------|--|------------------|
| getpid() getppid() | PXFGETPID() PXFGETPPID() chmod(), PXFCHMOD() chown(), PXFCHOWN() getegid(), PXFGETEGID() geteuid(), PXFGETEUID() getgid(), PXFGETGID() getgrgid(), PXFGETGRGID() getgrnam(), PXFGETGRNAM() getgroups(), PXFGETGROUPS() getlogin(), PXFGETLOGIN() getpgrp(), PXFGETPGRP() getpwnam(), PXFGETPWNAM() getpwuid(), PXFGETPWUID() getuid(), PXFGETUID() setgid(), PXFSETGID() setpgid(), PXFSETPGID() setsid(), PXFSETSID() setuid(), PXFSETUID() umask(), PXFUMASK() utime(), PXFUTIME() | |

Table K-8. POSIX Calls: Miscellaneous

| Toolkit Routines | Prohibited Routines | Allowed Routines |
|------------------|-----------------------------|-----------------------------|
| | localeconv() setlocale() | localeconv() setlocale() |

Table K-9. POSIX Calls: Terminal I/O

| Toolkit Routines | Prohibited Routines | Allowed Routines |
|------------------|---|------------------|
| | cfgetispeed(), PXF...() cfgetospeed(), PXF...() cfsetispeed(), PXF...() cfsetospeed(), PXF...() ctermid(), PXFCTERMID() getchar() gets() isatty(), PXFISATTY() printf() putchar() puts() scanf() tc...(), PXFTC...() ttyname(), PXFTTYNAME() | |

Table K-10. POSIX Calls: Status

| Toolkit Routines | Prohibited Routines | Allowed Routines |
|---|---------------------|--|
| fstat(), PXXFFSTAT() stat(), PXXFSTAT() uname(), PXXFUNAME() PXXFIS...() | | sysconf(), PXXFSYSCONF() times(), PXXFTIMES() |

Table K-11. POSIX Calls: FORTRAN77 Language Library

| Toolkit Routines | Prohibited Routines | Allowed Routines |
|---|---------------------|--|
| open() close() read(5,...) read(*,...) write(6,...) write(*,...) | | PXXFCALLSUBHANDLE() IPXXFCONST() PXXFCONST() PXXFGETSUBHANDLE() PXXFISCONST() PXXFSTRUCTCOPY() PXXFSTRUCTCREATE() PXXFSTRUCTFREE() PXXFSUBHANDLE()PXF<TYPE> GET() PXF<TYPE>SET() PXXFA<TYPE>GET() PXXFA<TYPE>SET() PXXFE<TYPE>GET() PXXFE<TYPE>SET() |

This page intentionally left blank.

Appendix L. Ephemeris And Attitude File Formats

WARNING: These are internal ECS file formats, these formats are NOT determined or defined by the Toolkit and are subject to change (the user interface to these files as defined by Toolkit APIs are not, however, subject to change). The following information is provided solely for those users interested in generating their own spacecraft ephemeris and attitude files for simulation purposes (the Toolkit otherwise provides a utility, *orbsim*, for this purpose—see section 6.2.6.1 Orbit and Attitude Simulator).

L.1 Spacecraft Ephemeris File Format

The ephemeris file is organized into 4 sections:

- 1) Fixed length header
- 2) Variable length header
- 3) Ephemeris records
- 4) Orbit metadata

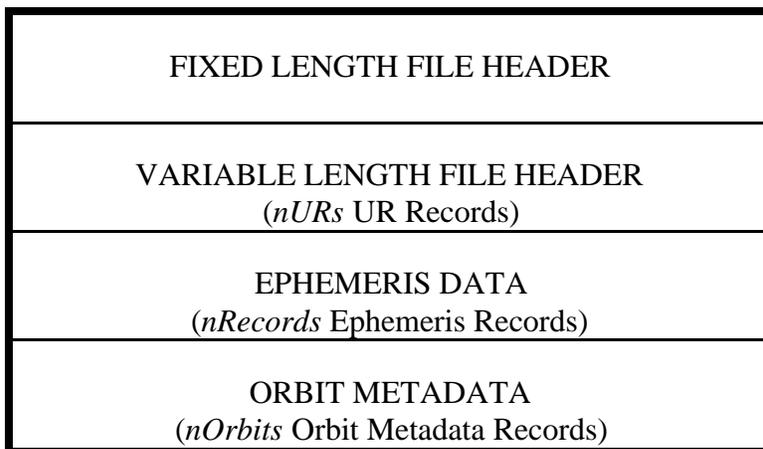


Figure L-1. Ephemeris File Schematic

Table L-1. Ephemeris File Fixed Length Header

| Type | Name | Meaning |
|---------------|--------------------|--|
| char | spacecraftID[24]* | Spacecraft Name (must be one of: "TRMM", "EOSAM1", "EOSPM1") |
| char | asciiTimeRange[48] | Start stop times to nearest hour or better, in ASCII |
| char | source[32] | Source of the data |
| char | version[8] | Version number (default = "1") |
| PGSt_double | startTime* | Ephemeris dataset start time, secTAI93 |
| PGSt_double | endTime* | Ephemeris dataset end time, secTAI93 |
| PGSt_real | interval | Expected interval between records, SI seconds |
| PGSt_uinteger | nURs* | Number of input dataset universal references |
| PGSt_uinteger | nRecords* | Number of ephemeris records |
| PGSt_uinteger | nOrbits** | Number of orbits spanned, including fragments |
| PGSt_uinteger | orbitNumberStart** | Number of first orbit or part orbit in file |
| PGSt_uinteger | orbitNumberEnd** | Number of last orbit or part orbit in file |
| char | keplerRefFrame[8] | Reference Frame (e.g. "TOD", "TOR" or "J2000") of the Keplerian Elements |
| PGSt_double | keplerElements[6] | Osculating Keplerian elements at epoch |
| PGSt_double | keplerEpochTAI | TAI 93 Epoch of the Reference Frame |
| PGSt_real | qaParameters[16] | Ephemeris data quality processing parameters |
| PGSt_real | qaStatistics[4] | Quality assurance statistics |
| char | spares[216] | Spares |

Notes

Fields marked with an asterisk (*) are critical and MUST be properly populated to ensure correct functioning of ALL Toolkit routines that rely on ephemeris data. Fields marked with a double asterisk (**) are critical to the operation of the Toolkit function PGS_EPH_GetEphMet() and must be properly populated for that tools to function correctly. None of the other fields is actually examined by the Toolkit (the fields MUST be included in the header but the values are not significant to the Toolkit).

Table L-2. Ephemeris File Universal Reference Record

| Type | Name | Meaning |
|------|----------------|--|
| char | parentURs[256] | Universal references of input datasets |

Notes

The number of these records (following the fixed length header) in the file MUST be equal to the value of nURs specified in the fixed length header (see Table L-1, Ephemeris File Fixed Length Header, above). The value of this record is not significant to the Toolkit.

Table L-3. Ephemeris Data Record

| Type | Name | Meaning |
|---------------|-----------|--|
| PGSt_double | time | Date and time as seconds from 1-1-93, secTAI93 |
| PGSt_double | xPos | X component of position vector, meters |
| PGSt_double | yPos | Y component of position vector, meters |
| PGSt_double | zPos | Z component of position vector, meters |
| PGSt_double | xVel | X component of velocity vector, meters/second |
| PGSt_double | yVel | Y component of velocity vector, meters/second |
| PGSt_double | zVel | Z component of velocity vector, meters/second |
| PGSt_uinteger | qFlagEph | Ephemeris data quality flag |
| char | spares[4] | Spares |

Notes

The number of these records (following the variable length header) in the file MUST be equal to the value of nRecords specified in the fixed length header (see Table L-1, Ephemeris File Fixed Length Header, above). The value of the variable “spares” is not significant to the Toolkit. See notes on data quality flags below (L.3 Quality Flags).

Table L-4. Ephemeris Orbit Metadata Record

| Type | Name | Meaning |
|---------------|-----------------------|--|
| PGSt_uinteger | orbitNumber | Orbit number, from beginning of mission |
| char | spares[4] | Spare |
| PGSt_double | orbitAscendTime; | Time of upward TOD equator crossing, secTAI93 |
| PGSt_double | orbitDescendTime | Time of downward TOD equator crossing, secTAI93 |
| PGSt_double | orbitDescendLongitude | Orbit down-crossing terrestrial longitude, radians |

Notes

The number of these records (following the ephemeris data records) in the file MUST be equal to the value of nOrbits specified in the fixed length header (see Table L-1, Ephemeris File Fixed Length Header, above). The value of the variable “spares” is not significant to the Toolkit.

L.2 Spacecraft Attitude File Format

The attitude file is organized into 3 sections:

- 1) Fixed length header
- 2) Variable length header
- 3) Attitude records

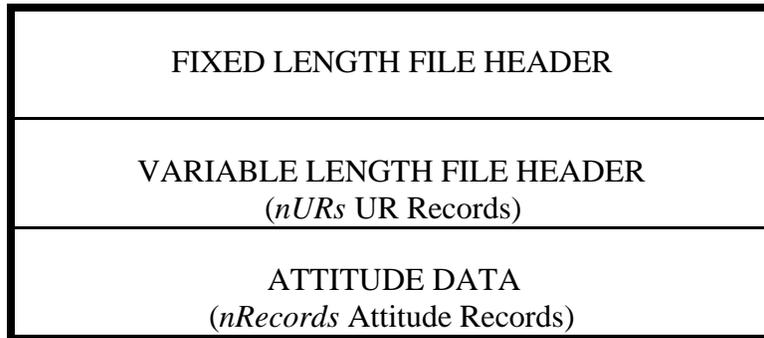


Figure L-2. Attitude File Schematic

Table L-5. Attitude File Fixed Length Header

| Type | Name | Meaning |
|---------------|--------------------|--|
| char | spacecraftID[24]* | Spacecraft Name (must be one of: "TRMM", "EOSAM1", "EOSPM1") |
| char | asciiTimeRange[48] | Start stop times to nearest hour or better, in ASCII |
| char | source[32] | Source of the data |
| char | version[8] | Version number (default = "1") |
| PGSt_double | startTime* | Ephemeris dataset start time, secTAI93 |
| PGSt_double | endTime* | Ephemeris dataset end time, secTAI93 |
| PGSt_real | interval | Expected interval between records, SI seconds |
| PGSt_uinteger | nURs* | Number of input dataset universal references |
| PGSt_uinteger | nRecords* | Number of ephemeris records |
| PGSt_uinteger | eulerAngleOrder[3] | Order of rotations as a permutation of 1=x, 2=y, 3=z |
| PGSt_real | qaParameters[16] | Attitude data quality processing parameters |
| PGSt_real | qaStatistics[4] | Quality assurance statistics |
| char | spares[280] | Spares |

Notes

Fields marked with an asterisk (*) are critical and MUST be properly populated to ensure correct functioning of ALL Toolkit routines that rely on attitude data. None of the other fields are actually examined by the Toolkit (the fields MUST be included in the header but the values are not significant to the Toolkit).

Table L-6. Attitude File Universal Reference Record

| Type | Name | Meaning |
|------|----------------|--|
| char | parentURs[256] | Universal references of input datasets |

Notes

The number of these records (following the fixed length header) in the file MUST be equal to the value of nURs specified in the fixed length header (see Table L-5, Attitude File Fixed Length Header, above). The value of this record is not significant to the Toolkit.

Table L-7. Attitude Data Record

| Type | Name | Meaning |
|---------------|------------------|--|
| PGSt_double | time | Date and time as seconds from 1-1-93, secTAI93 |
| PGSt_double | firstEulerAngle | First Euler angle, radians. |
| PGSt_double | secondEulerAngle | Second Euler angle, radians. |
| PGSt_double | thirdEulerAngle | Third Euler angle, radians. |
| PGSt_double | xRate | Angular rate about body X axis, radians/s |
| PGSt_double | yRate | Angular rate about body Y axis, radians/s |
| PGSt_double | zRate | Angular rate about body Z axis, radians/s |
| PGSt_uinteger | qFlagAtt | Attitude data quality flag |
| char | spares[4] | Spares |

Notes

The number of these records (following the variable length header) in the file MUST be equal to the value of nRecords specified in the fixed length header (see Table L-5, Attitude File Fixed Length Header, above). The value of the variable “spares” is not significant to the Toolkit. See notes on data quality flags below (L.3 Quality Flags).

L.3 Quality Flags

Quality flags definitions for ephemeris and attitude are *suggested* here. In actual cases, the flags bits may be set according to spacecraft-criteria which should be explained and supported with references to original documents. Tables L-8 and L-9 show suggested usage for platform generic and platform specific quality flags. In a specific case, not all fields may be populated. For table 2B the usage could be quite different, for different spacecraft, but we have requested that bit 16 be reserved for a platform-specific "fatal" flag, *if* the data provider intends to send data packets considered to be quite unreliable (the alternative is to send no data in such cases). The SDP Toolkit tool for ephemeris and attitude access are user-callable, but are also used by higher level tools. The user interface differs somewhat in the two cases. When the access tool is called directly, it passes the flags on to the user. *In the other case, for example if the user is accessing geolocation services, the interface has to be different, because the user has not requested the flags. In recent Toolkit releases, an error is returned only when large data gaps exist; the flags were ignored. Work in progress will implement fuller recognition of quality flags by higher level tools*

that call the ephemeris tool. Thus, both missing data and bad quality data will result in warning or error messages. The current plan is to implement, as default behavior, data rejection when bit 16 is set. We also intend to enable the user to set a quality flag mask, if desired, in the Process Control File, enabling rejection based on other bits of her or his own choice. For this reason, data providers are encouraged to establish practical definitions of flag bits suitable for users to check questionable points. In particular, bits 2, 5, 6 and 9, if set, might in the future be used by users to reject points. These represent the large gap and red variation limits. It is generally supposed that some range or continuity checks have been imposed on the data, and will be reflected in some of the flags ("yellow" and "red" limits exceeded). Since the checks could be range or continuity checks, accompanying documentation should explain the procedure, i.e. the meaning of these limits. So that the parameters used in checking will be available in the data sets themselves. We are recommending that the Red limit bit be set so as to statistically reject not more than 0.01% of the data when the variation is normal statistics, and the Yellow limit be set so as to reject not more than 0.1%.

In reading Tables L8 and L9, please bear in mind that more detail has been included in the quality data for attitude in the examples than for ephemeris, because our first program spacecraft, TRMM, will be providing us smoothed, fitted ephemeris data and less refined attitude data. For other platforms, it might prove advisable to include more detail in the ephemeris quality flags.

Table L-8. Quality Flags - Platform Generic

| Bit | Bit Assignment | Description |
|-------|--------------------------------|---|
| 0 | Overall Quality Summary | Set if any quality check is failed; unset for ideal data. Data point can still be useful even if this bit is set; scrutiny of the other bits would be required however. Bits 1 and 16 are unset in this instance of ideal data. |
| 1 | Data State Summary | Set if <i>any</i> generic data quality bit is set (bits 2 - 11) |
| 2 | Red Limit Low Exceeded | Low red limit has been exceeded. |
| 3 | Yellow Limit Low Exceeded | Low yellow limit has been exceeded. |
| 4 | Yellow Limit High Exceeded | High yellow limit has been exceeded. |
| 5 | Red Limit High Exceeded | High red limit has been exceeded. |
| 6 | Long Data Gap Follows | A significant data gap originally followed this data point. |
| 7 | Short Data Gap Follows | A minor data gap originally followed this data point. |
| 8 | Short Data Gap Precedes | A minor data gap originally preceded this data point. |
| 9 | Long Data Gap Precedes | A significant data gap originally preceded this data point. |
| 10 | Point is a repaired data point | Used for points inserted by software <i>prior to Toolkit</i> (interpolated). |
| 11 | <i>Quality flag problem</i> | <i>quality data not available (bits 0-5 not meaningful)</i> |
| 12-15 | Unassigned | Reserved for SDP Toolkit use. |

Notes

Bits 1-15 are Platform Generic Flags are for general data quality flagging, and are intended to apply to all platforms and both attitude and ephemeris data. Bit 0 is least significant.

Table L-9. Example of Attitude Quality Flags - TRMM Platform Specific

| Bit | Bit Assignment | Description |
|-------|-------------------------------------|--|
| 16 | <i>Platform Specific Fatal Flag</i> | <i>Set if any fatal platform specific quality bit is set</i> |
| 17 | QAC Flag | Data transmission flagged in QAC list. |
| 18 | ACS State Flag | ACS state incompatible with attitude acquisition. |
| 19 | Yaw Maneuver | Set if ACS yaw maneuver in progress. |
| 20 | Yaw Update Flag | Set if the ACS is awaiting a yaw update. The ACS is in transition between a delta-v, delta-h or inertial hold maneuver and nominal mode. The attitude is likely good but not guaranteed. |
| 21 | Contingency Mode Flag | Set if the ACS is operating in a degraded state due to an Earth sensor failure. |
| 22 | Inertial Hold Flag | Spacecraft is flying in inertial space locked mode. |
| 23-31 | Unassigned | Available for other platform specific data—quality or other. |

Notes

Bits 17 through 31 are Platform Specific Flags reserved for data flagging except that bit 16 is common to all platforms. Bit 31 is *most significant*. The definitions outlined here are for the TRMM platform.

This page intentionally left blank.

Appendix M. Problem Identification List

The list of known problems as of 11/15/96 for Toolkit 5.1.1 delivery of the SDP Toolkit can be found in section 5 of the SDP Toolkit 5.1.1 Version Description Document (VDD) for the ECS Project

This page intentionally left blank.

Abbreviations and Acronyms

| | |
|-------|--|
| A.A. | Astronomical Almanac |
| AA | ancillary data access |
| AI&T | algorithm integration & test |
| AIRS | Atmospheric Infrared Sounder |
| API | application program interface |
| APID | application process identifier |
| ASTER | Advanced Spaceborne Thermal Emission and Reflection Radiometer (formerly ITIR) |
| BNF | Bachus–Nauer Form |
| CBP | celestial body position |
| CCR | configuration change request |
| CCSDS | Consultative Committee on Space Data Systems |
| CDRL | Contract Data Requirements List |
| CDS | CCSDS day segmented time code |
| CERES | Clouds and Earth Radiant Energy System |
| CM | configuration management |
| COTS | commercial off–the–shelf software |
| CRC | cyclic redundancy code |
| CSC | coordinate system conversion |
| CSMS | Communications and Systems Management Segment (ECS) |
| CUC | constant and unit conversions |
| CUC | CCSDS unsegmented time code |
| DAAC | distributed active archive center |
| DBMS | database management system |
| DCE | distributed computing environment |
| DCW | Digital Chart of the World |

| | |
|--------|---|
| DDF | data distribution facility (Pacor) |
| DEM | digital elevation model |
| DPFT | Data Processing Focus Team |
| DTM | digital terrain model |
| ECI | Earth centered inertial |
| ECR | Earth centered rotating |
| ECS | EOSDIS Core System |
| EDC | Earth Resources Observation Systems (EROS) Data Center |
| EDHS | ECS Data Handling System |
| EDOS | EOSDIS Data and Operations System |
| EOS | Earth Observing System |
| EOSAM | EOS AM Project (morning spacecraft series) |
| EOSDIS | Earth Observing System Data and Information System |
| EOSPM | EOS PM Project (afternoon spacecraft series) |
| EPH | ephemeris data access |
| ESDIS | Earth Science Data and Information System (GSFC Code 505) |
| ET | ephemeris tool |
| FDF | flight dynamics facility |
| FNOC | Federal Naval Operations Center |
| FOV | field of view |
| ftp | file transfer protocol |
| GAST | Greenwich apparent sidereal time |
| GCT | geo-coordinate transformation |
| GCTP | general cartographic transformation package |
| GIS | geographic information systems |
| GMST | Greenwich mean sidereal time |
| GPS | Global Positioning System |
| GSFC | Goddard Space Flight Center |
| HDF | hierarchical data format |

| | |
|--------|--|
| HITC | Hughes Information Technology Corporation |
| HOM | Hotine Oblique Mercator |
| http | hypertext transport protocol |
| I&T | integration & test |
| I/O | input/output |
| IAU | International Astronomical Union |
| ICD | interface control document |
| IDL | interactive data language |
| IEEE | Institute of Electrical and Electronic Engineers |
| IERS | International Earth Rotation Service |
| IMS | information management system |
| IP | Internet protocol |
| IWG | Investigator Working Group |
| JNC | jet navigational charts |
| JPL | Jet Propulsion Laboratory |
| LaRC | Langley Research Center |
| LIS | Lightening Imaging Sensor |
| M&O | maintaince and operations |
| MCF | metadata configuration file |
| MDU | missing data unit |
| MDUE | Missing Data Unit Entry |
| MDUL | missing data unit list |
| MEM | memory management |
| MET | metadata |
| MODIS | Moderate-Resolution Imaging Spectroradiometer |
| MSFC | Marshall Space Flight Center |
| NASA | National Aeronautics and Space Administration |
| NCSA | National Center for Supercomputer Applications |
| netCDF | network common data format |

| | |
|-------|---|
| NGDC | National Geophysical Data Center |
| NMC | National Meteorological Center (NOAA) |
| ODL | object description language |
| PACOR | packet processor |
| PC | process control |
| PCF | process control file |
| PDPS | planning & data production system |
| PDR | Preliminary Design Review |
| PDS | production data set |
| PGE | product generation executive (formerly product generation executable) |
| PGS | Product Generation System |
| PGSTK | Product Generation System Toolkit |
| POSIX | Portable Operating System Interface for Computer Environments |
| QA | quality assurance |
| QAC | quality and accounting capsule |
| RDBMS | relation data base management system |
| RPC | remote procedure call |
| RRDB | recommended requirements database |
| SCF | Science Computing Facility |
| SDP | science data production |
| SDPF | science data processing facility |
| SDPS | Science Data Processing Segment (ECS) |
| SES | scheduling and execution subsystem |
| SFDU | standard formatted data unit |
| SGI | Silicon Graphics Incorporated |
| SI | systeme international |
| SMF | status message file |
| SMP | Symmetric Multi-Processor |
| SOM | Space Oblique Mercator |

| | |
|-------|--|
| SPCS | State Plane Coordinates Spheroid |
| SPSO | Science Processing Support Office |
| SSM/I | Special Sensor for Microwave/Imaging |
| TAI | International Atomic Time |
| TBD | to be determined |
| TD | time date conversion |
| TDB | Barycentric Dynamical Time |
| TDRSS | Tracking and Data Relay Satellite System |
| TDT | Terrestrial Dynamical Time |
| TLCF | team leader computing facility |
| TRMM | Tropical Rainfall Measuring Mission (joint US – Japan) |
| TSS | (TDRSS) Service Session |
| UARS | Upper Atmosphere Research Satellite |
| UCAR | University Corporation for Atmospheric Research |
| URL | universal reference locator |
| USDC | United States Department of Commerce |
| USNO | United States Naval Observatory |
| UT | universal time |
| UTC | Coordinated Universal Time |
| UTCf | universal time correlation factor |
| UTM | universal transverse mercator |
| VCDU | virtual channel data unit |
| VPF | vector product format |
| WWW | World Wide Web |